charRecog.R

karanYsingh

2021-03-31

```r
library(ramify)
```

```
## Warning: package 'ramify' was built under R version 4.0.4

##
## Attaching package: 'ramify'

## The following object is masked from 'package:graphics':
##
##     clip
```

```r
library(jpeg)
```

```
## Warning: package 'jpeg' was built under R version 4.0.3
```

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.0.3

## Loading required package: lattice

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 4.0.3
```

```r
library(OpenImageR)
```

```
## Warning: package 'OpenImageR' was built under R version 4.0.4
```

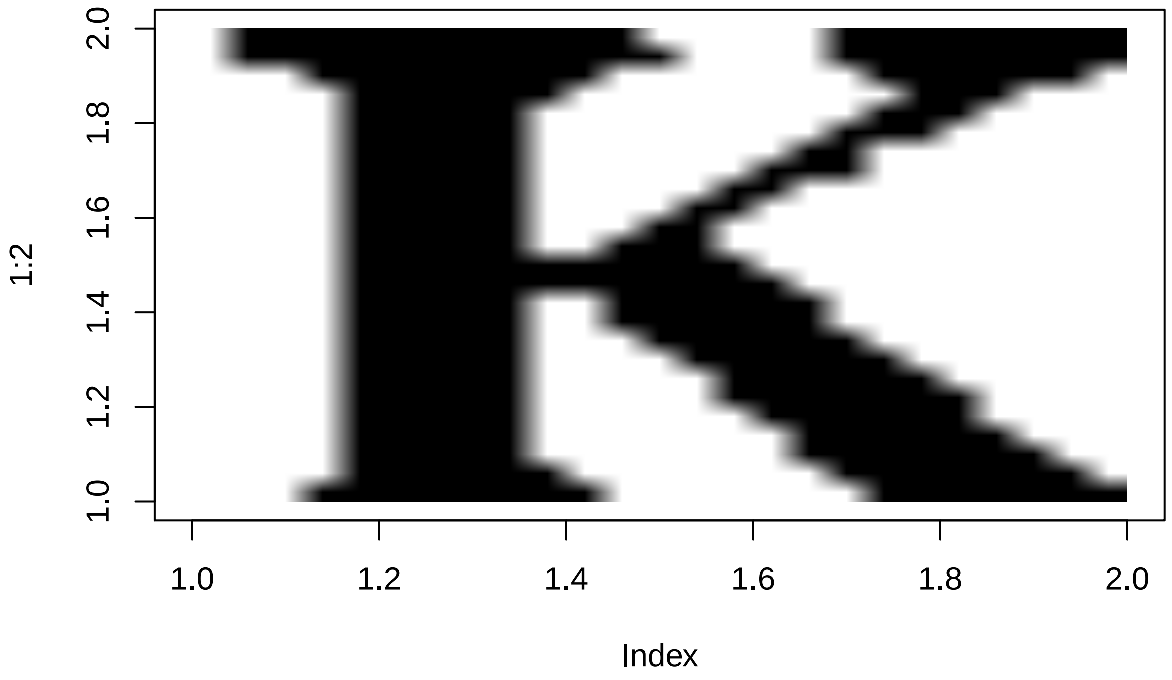
```r
##DIRECTORY CONTAINING IMAGES DOWNSCALED TO 100X100.
##EACH CLASS CONTAIN 13 Images of different fonts.
KPath = "C:/Users/91828/Documents/Rlab/CharRecog/Ks/"
NPath = "C:/Users/91828/Documents/Rlab/CharRecog/Ns/"
list.files(KPath)
```

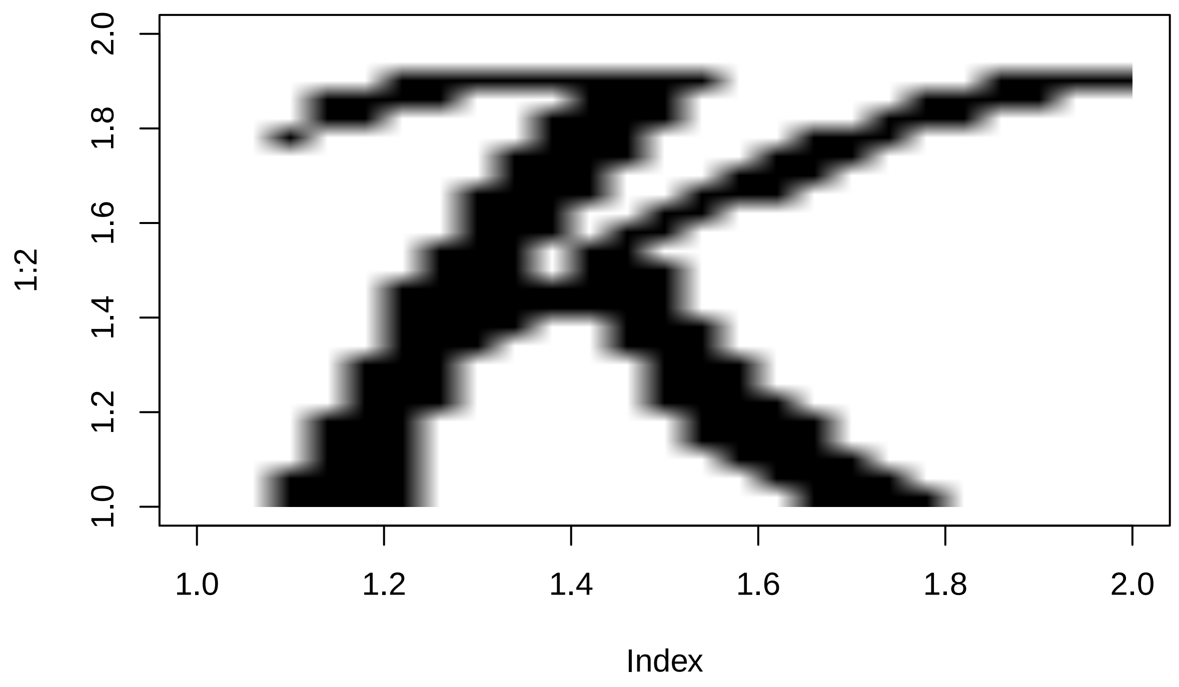
```
##  [1] "k1.jpg"  "k10.jpg" "k11.jpg" "k12.jpg" "k13.jpg" "k14.jpg" "k15.jpg"
##  [8] "k16.jpg" "k17.jpg" "k18.jpg" "k19.jpg" "k2.jpg"  "k20.jpg" "k3.jpg"
## [15] "k4.jpg"  "k5.jpg"  "k6.jpg"  "k7.jpg"  "k8.jpg"  "k9.jpg"
```

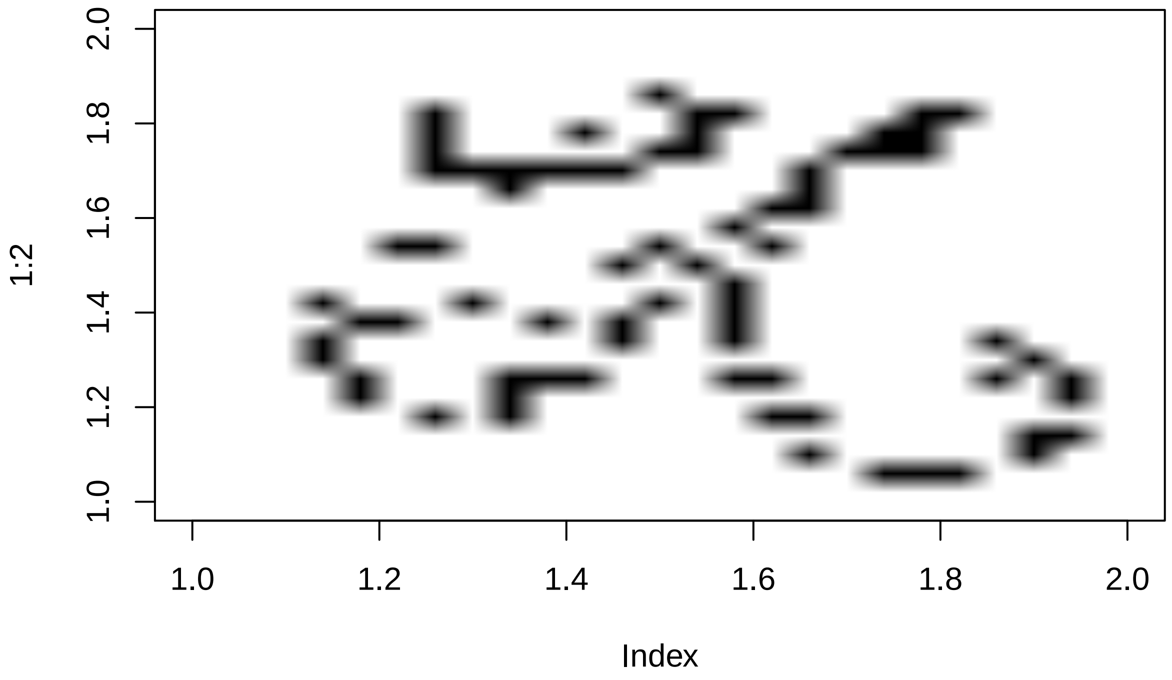
```r
list.files(NPath)
```

```
##  [1] "n1.jpg"  "n10.jpg" "n11.jpg" "n12.jpg" "n13.jpg" "n14.jpg" "n15.jpg"
##  [8] "n16.jpg" "n17.jpg" "n18.jpg" "n19.jpg" "n2.jpg"  "n20.jpg" "n3.jpg"
## [15] "n4.jpg"  "n5.jpg"  "n6.jpg"  "n7.jpg"  "n8.jpg"  "n9.jpg"
```

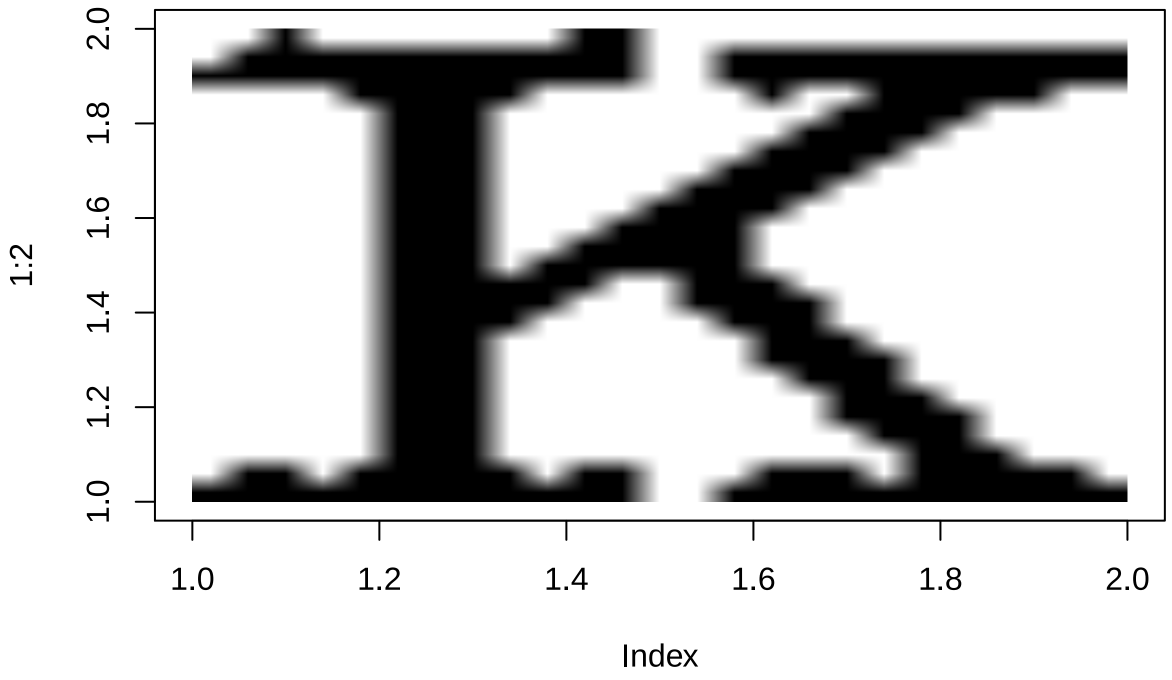
```r
options(error=function() dump.frames(to.file=TRUE))
##HELPER FUNCTION TO PLOT IMAGES OF DIRECTORY
plotImage <- function(tst){

  if(exists("rasterImage")){
       plot(1:2, type='n')
       rasterImage(tst,1,1,2,2)
  }
}

##INITIALISING DATA FRAME
dataset = data.frame()

##FILENAMES OF font K
Kfilenames = list.files(KPath)
# par(mfrow=c(4,4))
for(i in Kfilenames[1:length(Kfilenames)]){
  txt <- paste(KPath,i,sep="")

  #Read image
  tst <- readJPEG(txt)
  tst <- resizeImage(tst, w = 25, h = 25)

  #Converting values to 1 or 0
  tst <- ifelse(tst>0.5,1,0)

  #taking only Black pixel values
  #(since image is black and white taking any pixel value will do)
  tst <- tst[,,1]
  plotImage(tst)
  #flattening array from (100,100) to (10000,1)
  tst <- ramify::flatten(tst)

  #Adding label to the data, '1' for K and '0' for N.
  tst<-c(1,tst)

  #Adding rows to dataset
  dataset<- rbind(dataset,tst)
}
```

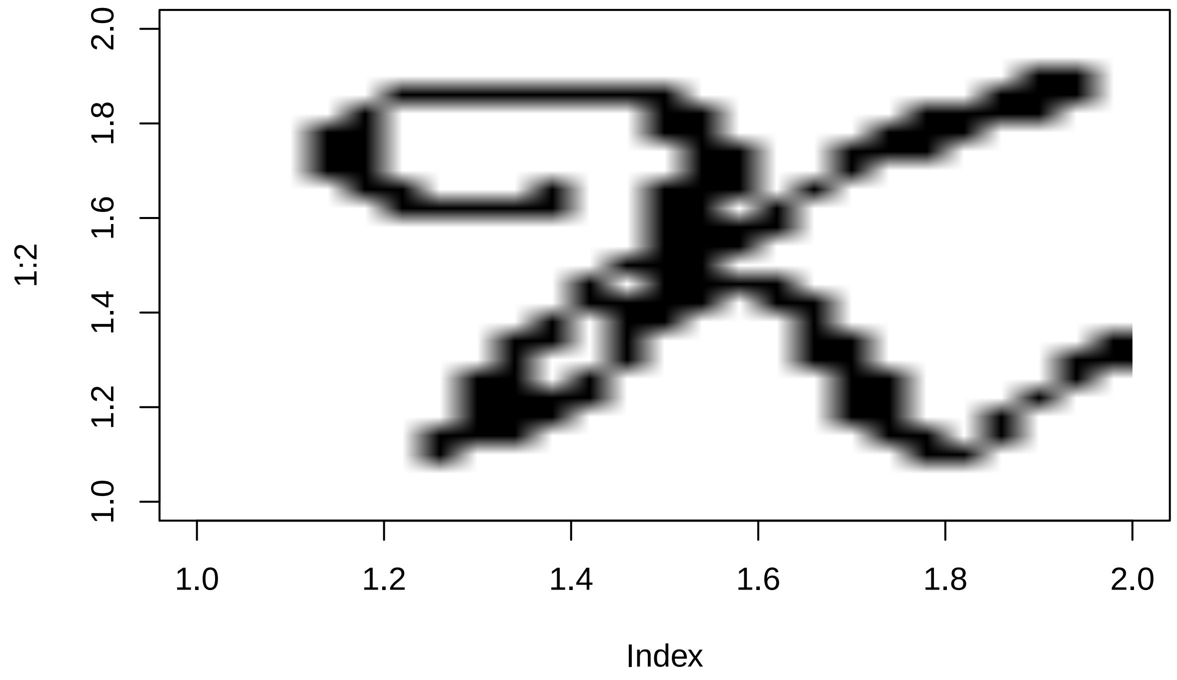

2.0
1.8
1.6
1.4
1.2
1.0
1:2
1.0
1.2
1.4
1.6
1.8
2.0
Index


1:2
Index
1.0
1.2
1.4
1.6
1.8
2.0

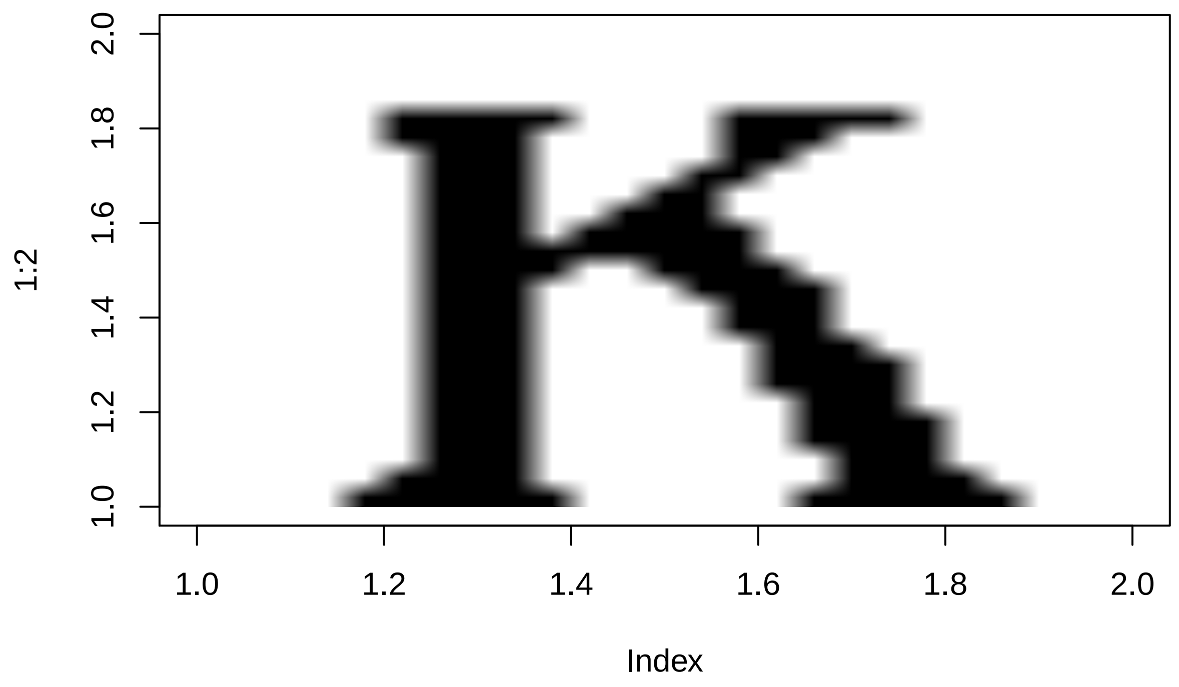


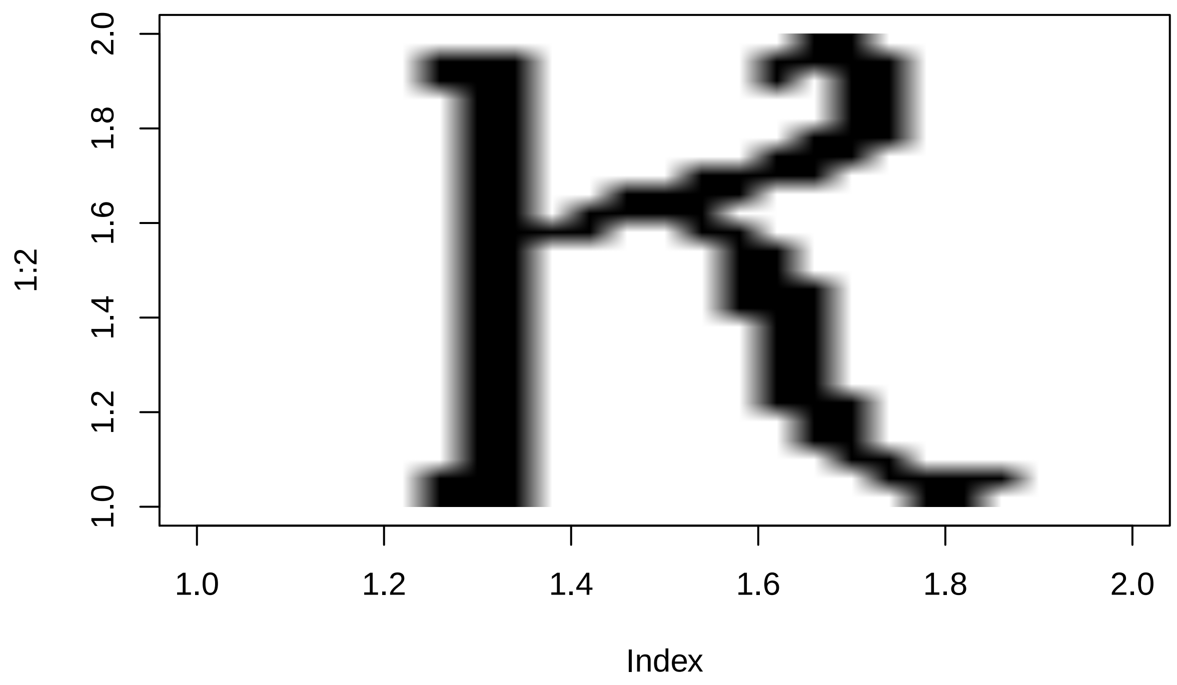
1:2
Index
1.0
1.2
1.4
1.6
1.8
2.0

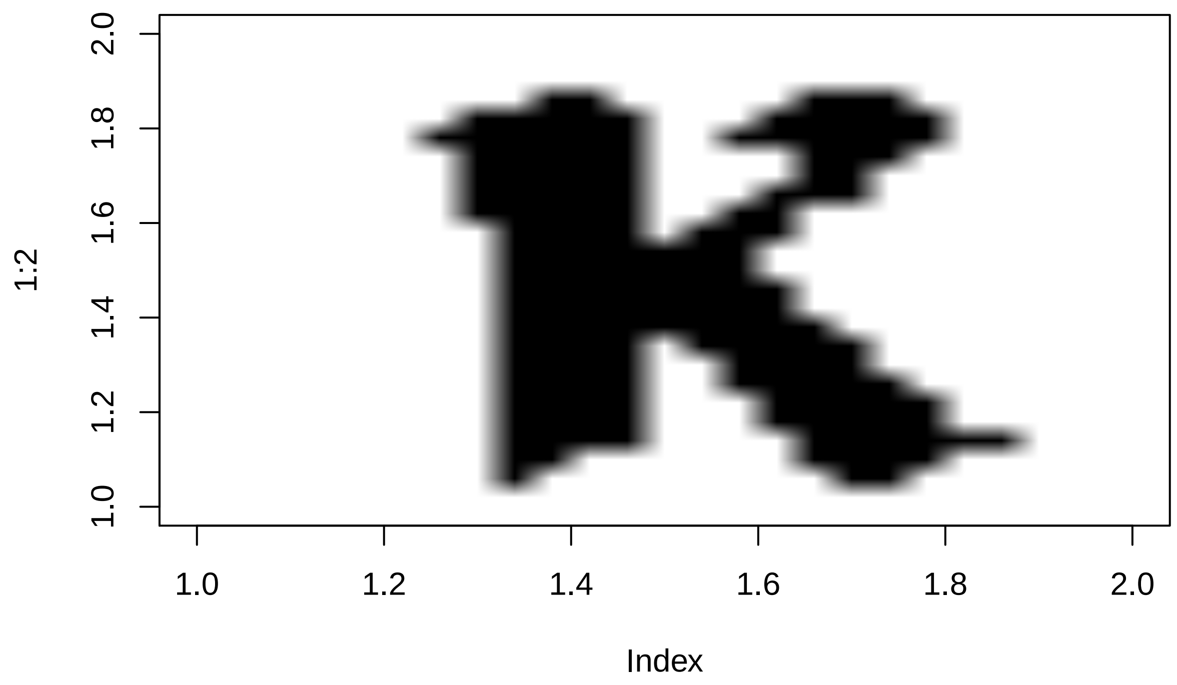

2.0
1.8
1.6
1.4
1.2
1.0
1:2
1.0
1.2
1.4
1.6
1.8
2.0
Index

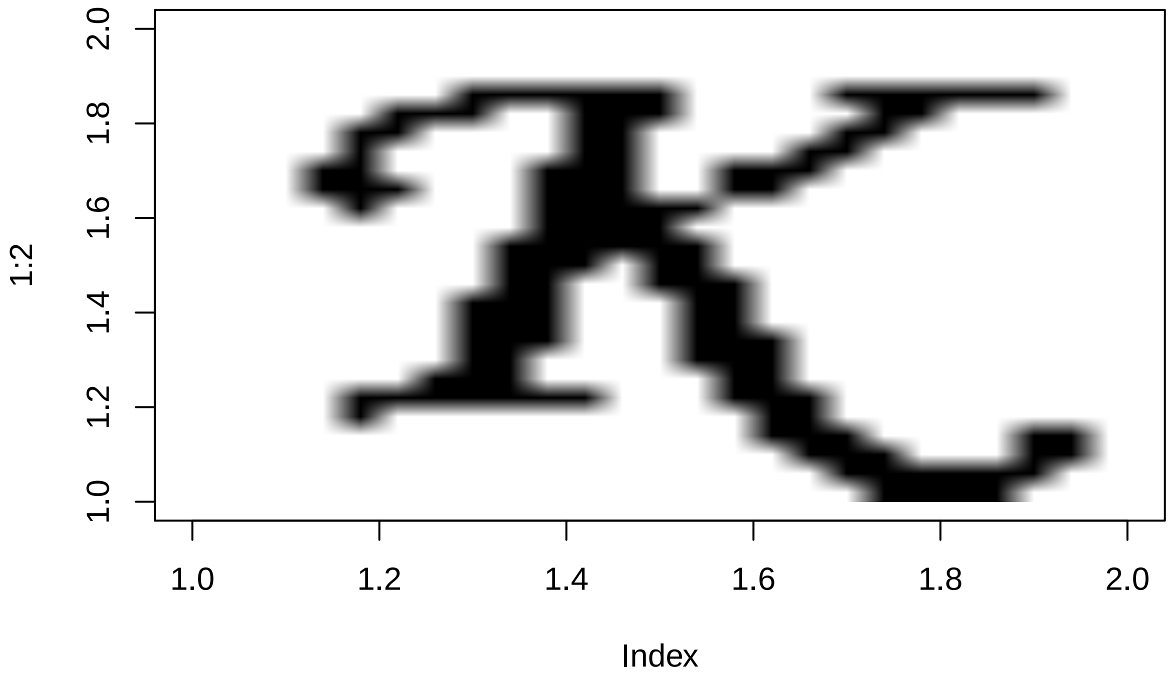


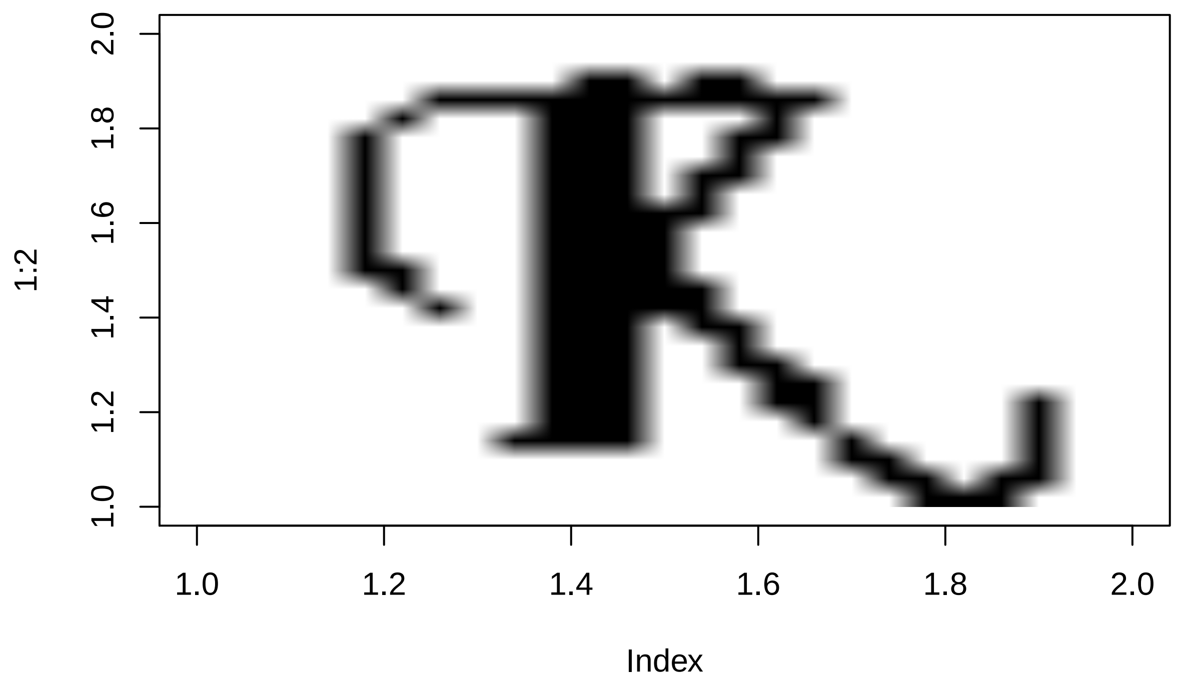
1:2
2.0
1.8
1.6
1.4
1.2
1.0
1.0
1.2
1.4
1.6
1.8
2.0
Index

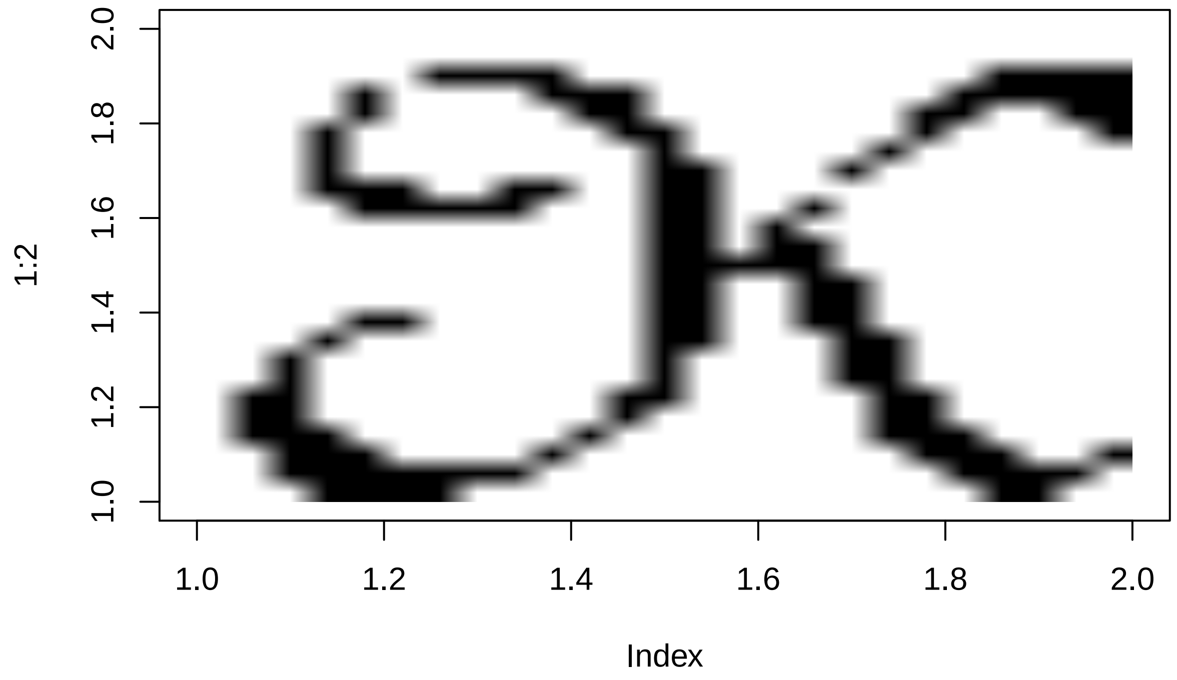

2.0
1.8
1.6
1.4
1.2
1.0
1:2
1.0
1.2
1.4
1.6
1.8
2.0
Index

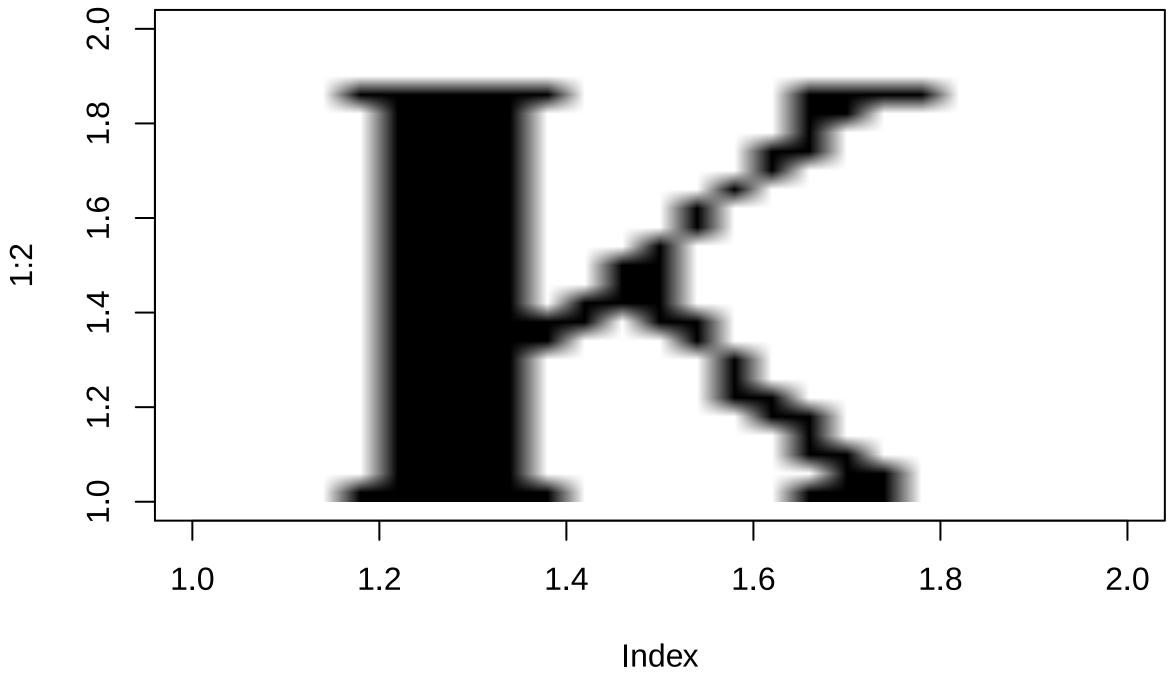

1:2
2.0
1.8
1.6
1.4
1.2
1.0
1.0
1.2
1.4
1.6
1.8
2.0
Index

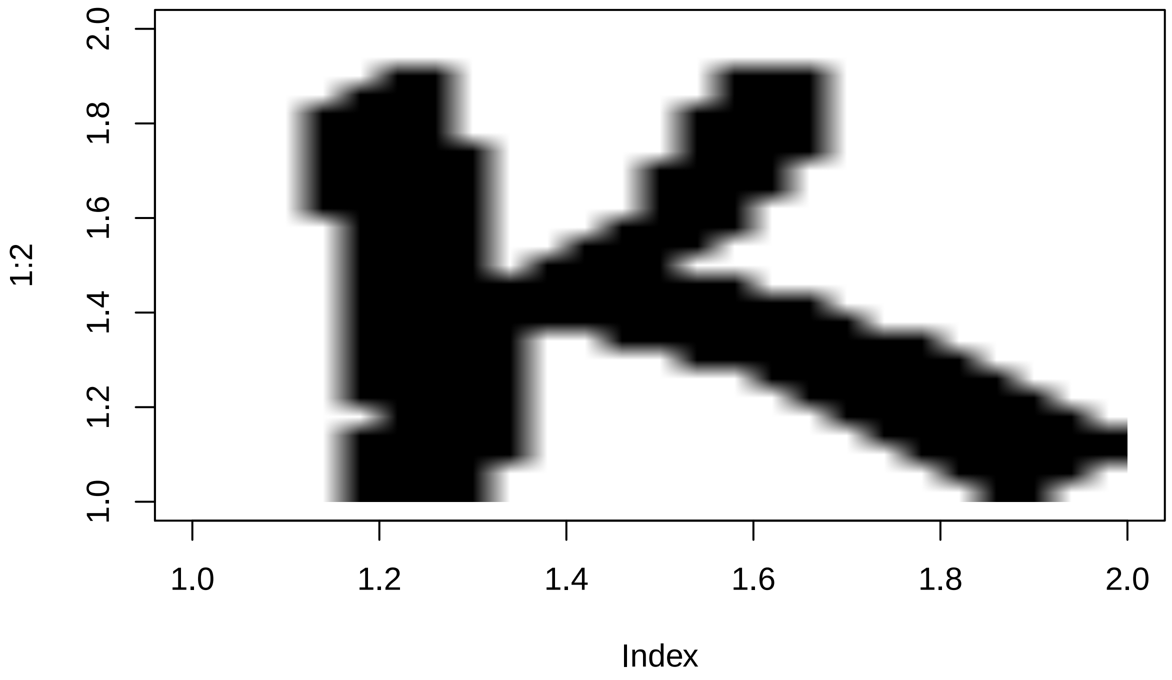

1:2
Index
1.0
1.2
1.4
1.6
1.8
2.0

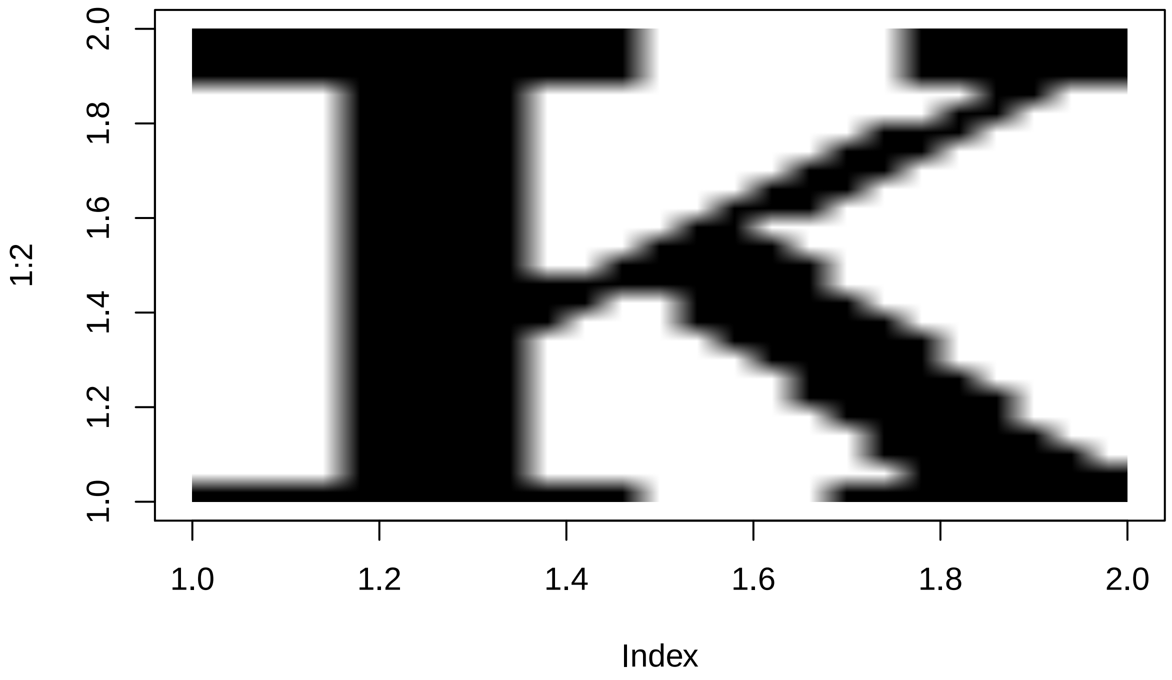

2.0
1.8
1.6
1.4
1.2
1.0
1:2
1.0
1.2
1.4
1.6
1.8
2.0
Index

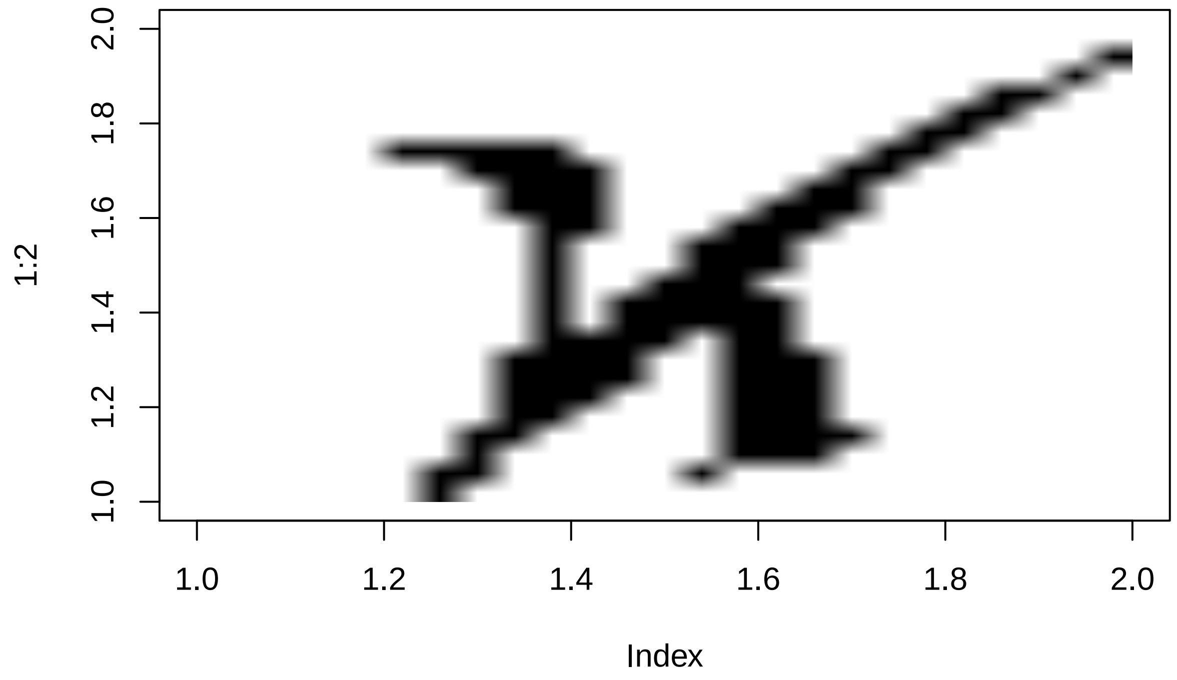

2.0
1.8
1.6
1.4
1.2
1.0
1:2
1.0
1.2
1.4
1.6
1.8
2.0
Index

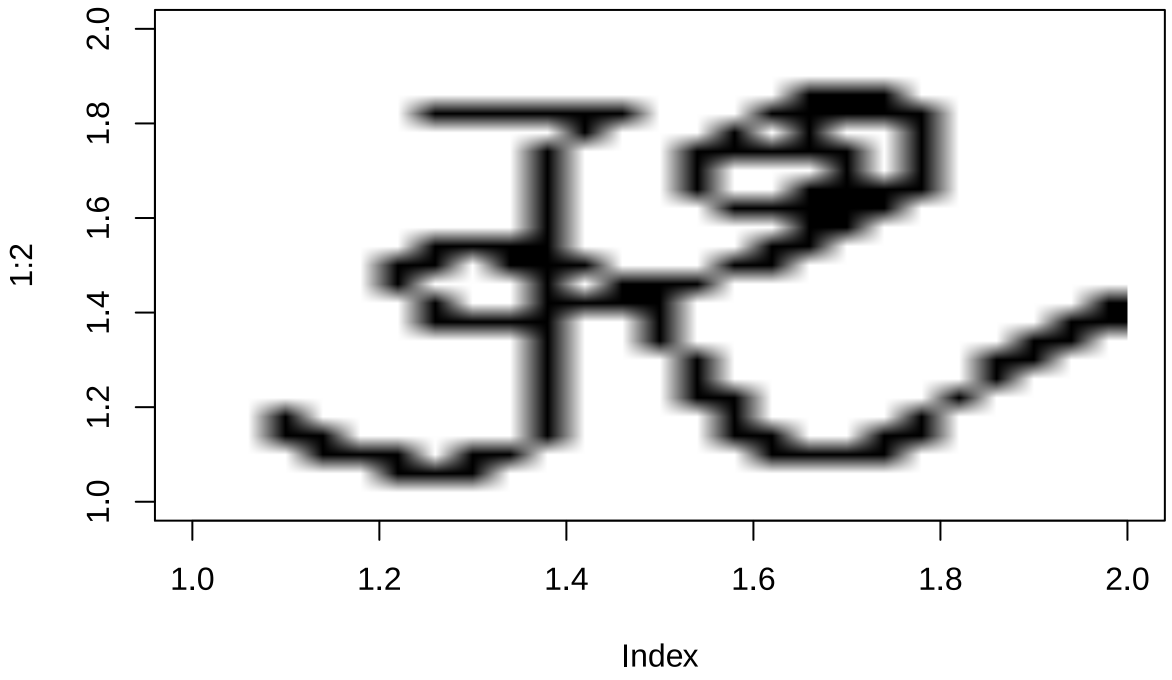

2.0
1.8
1.6
1.4
1.2
1.0
1:2
1.0
1.2
1.4
1.6
1.8
2.0
Index

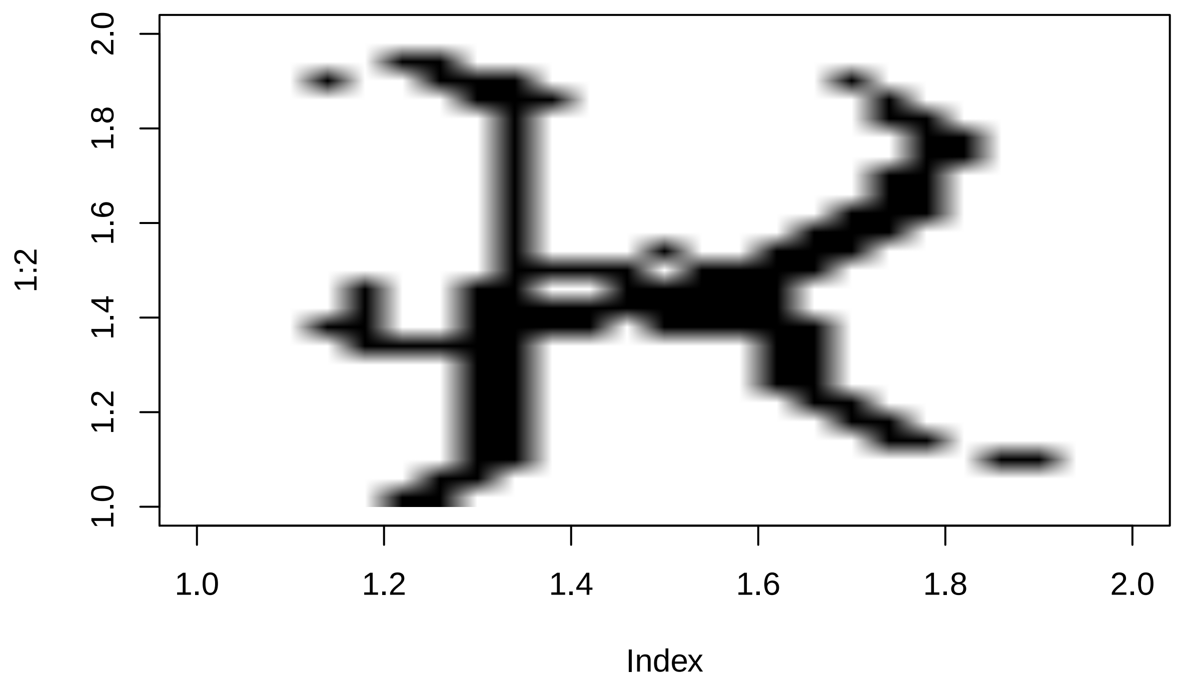

1:2
2.0
1.8
1.6
1.4
1.2
1.0
1.0
1.2
1.4
1.6
1.8
2.0
Index

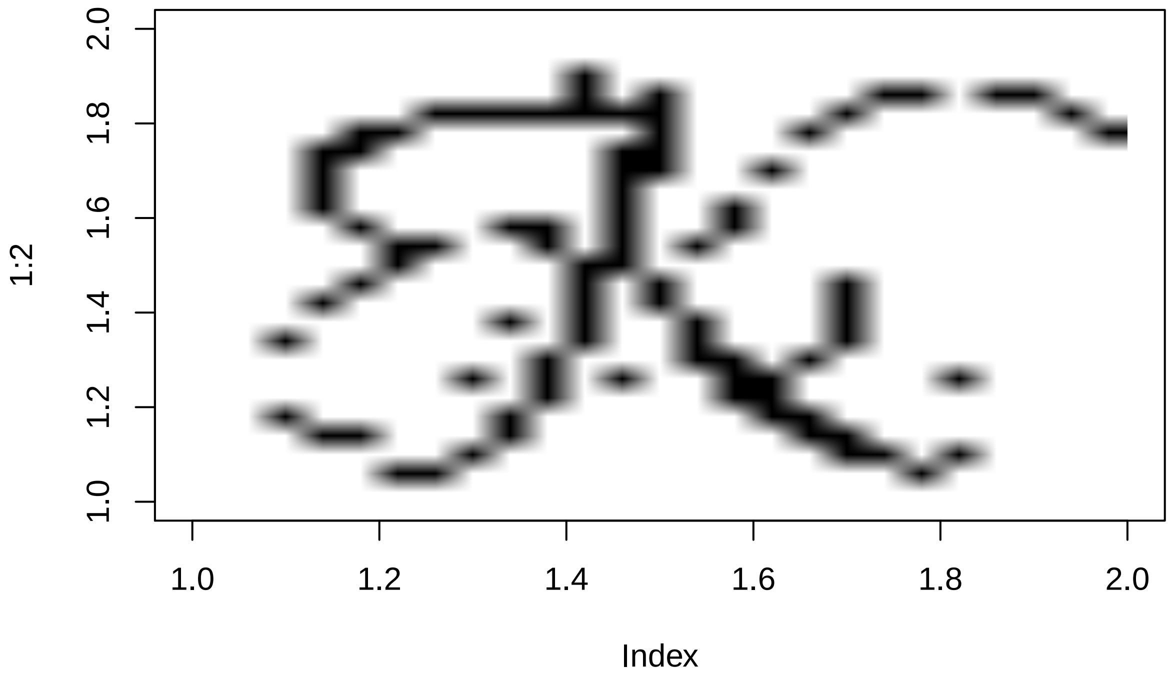

2.0
1.8
1.6
1.4
1.2
1.0
1:2
1.0
1.2
1.4
1.6
1.8
2.0
Index


1:2
2.0
1.8
1.6
1.4
1.2
1.0
1.0
1.2
1.4
1.6
1.8
2.0
Index

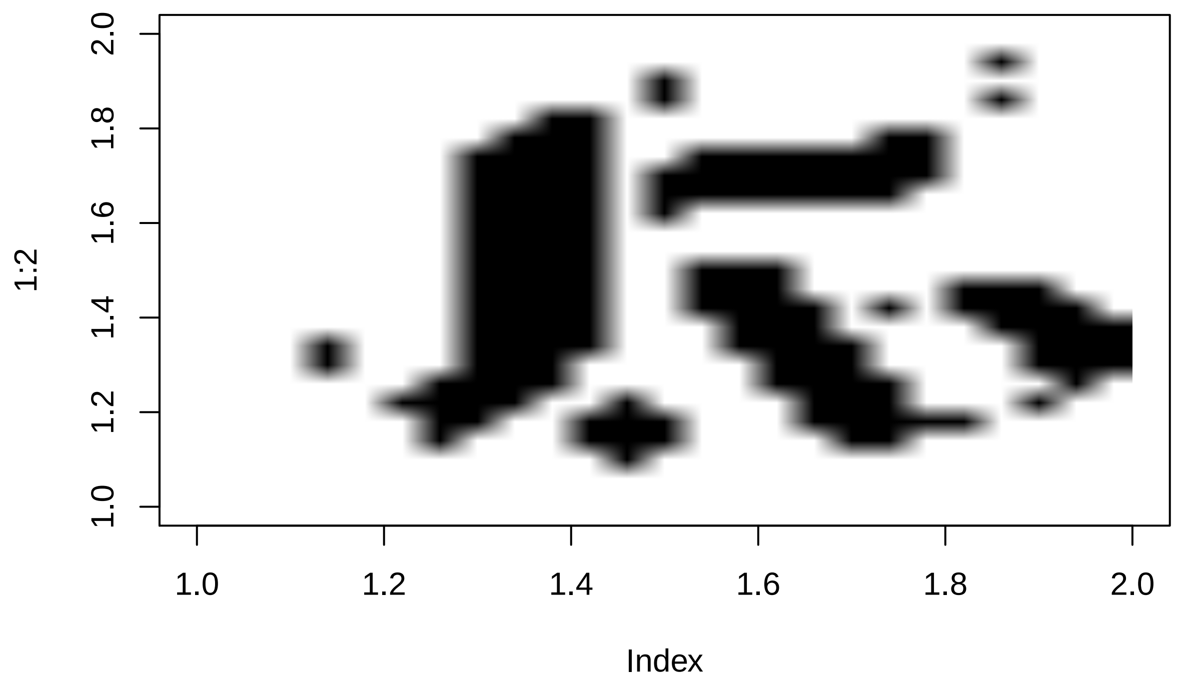

1:2
Index
1.0
1.2
1.4
1.6
1.8
2.0


2.0
1.8
1.6
1.4
1.2
1.0
1:2
1.0
1.2
1.4
1.6
1.8
2.0
Index

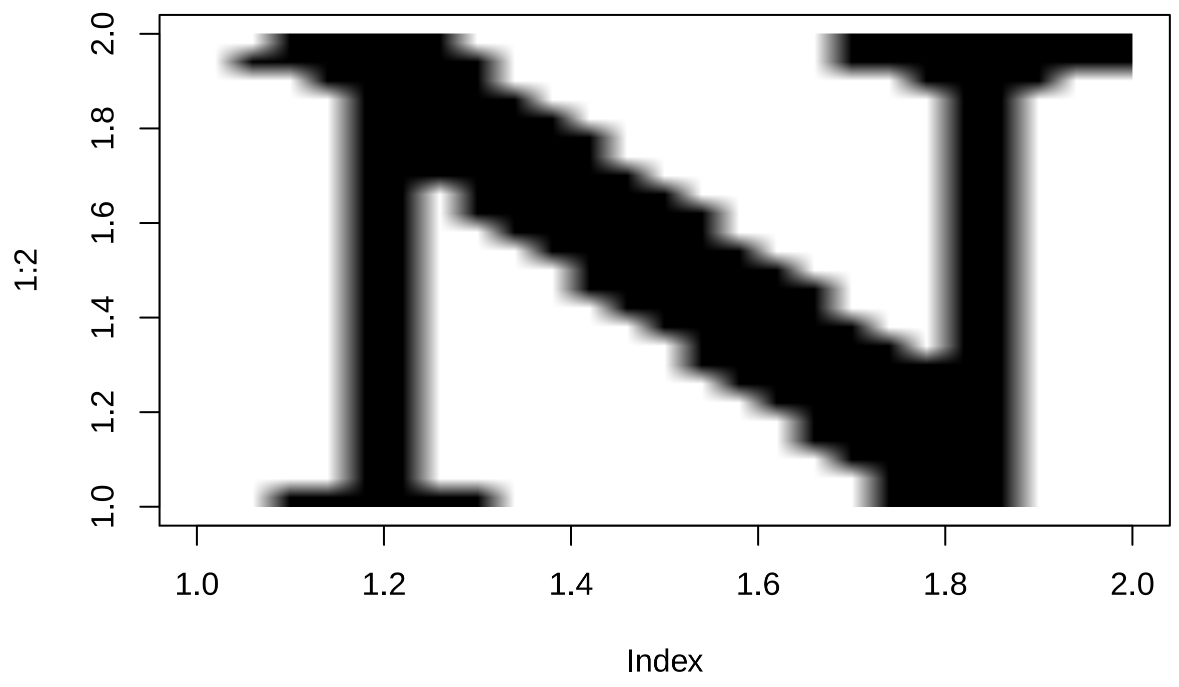

1:2
2.0
1.8
1.6
1.4
1.2
1.0
1.0
1.2
1.4
1.6
1.8
2.0
Index



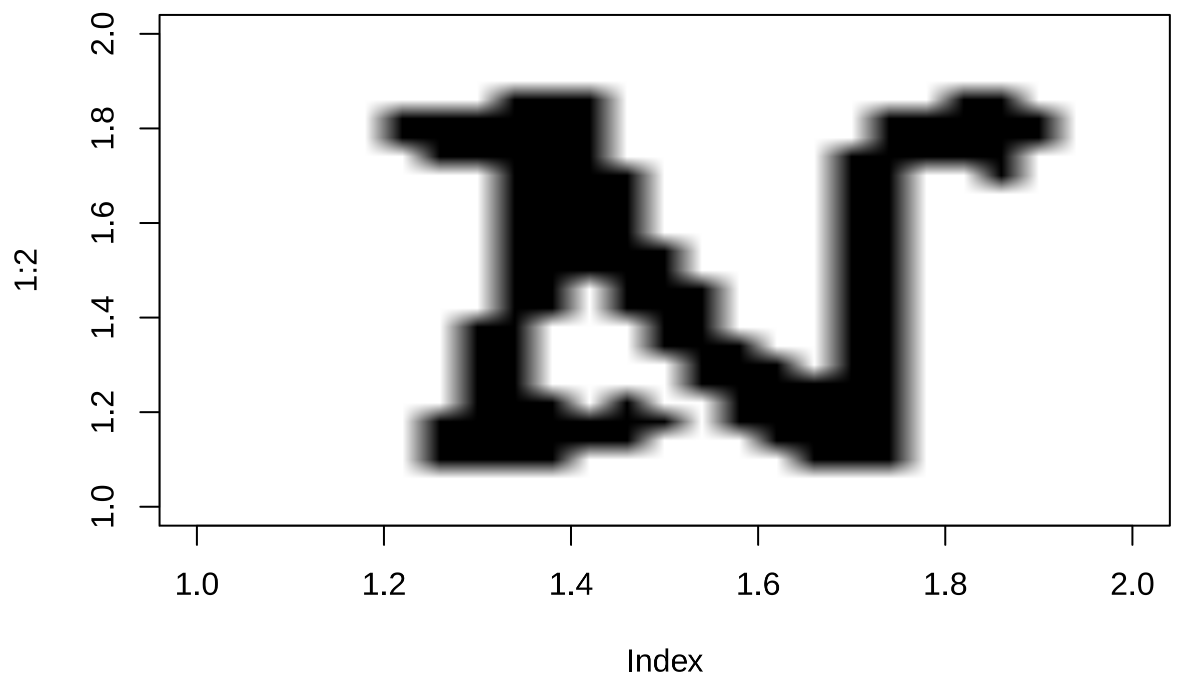
2.0
1.8
1.6
1.4
1.2
1.0
1:2
1.0
1.2
1.4
1.6
1.8
2.0
Index

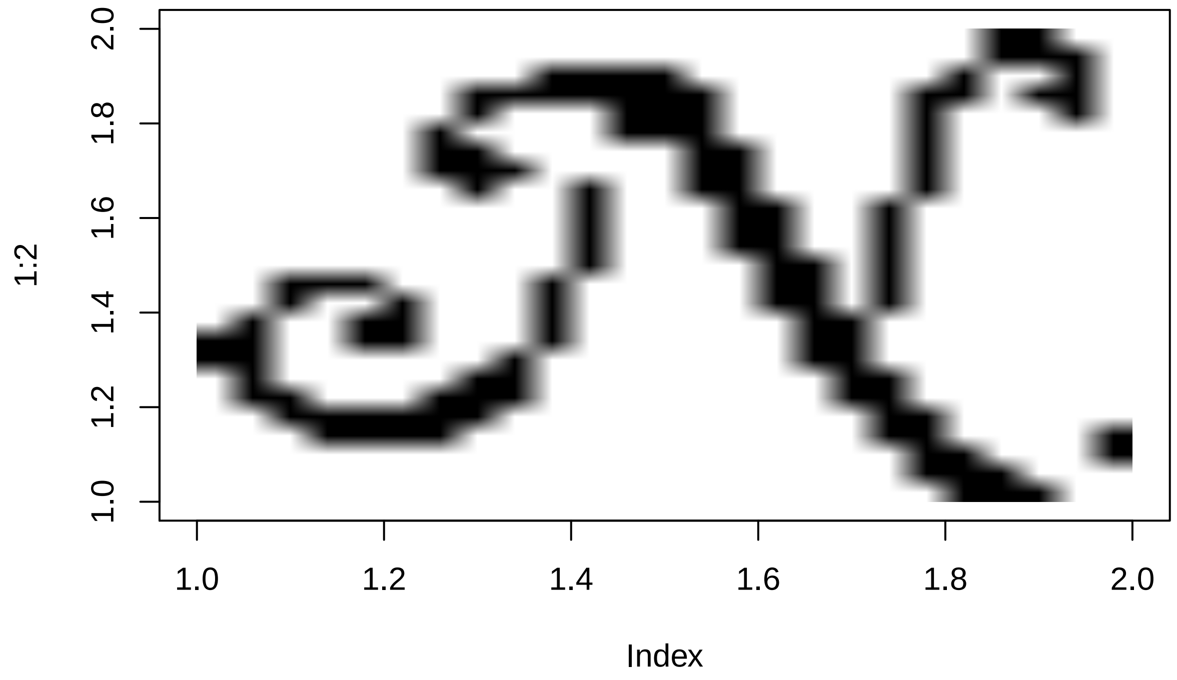


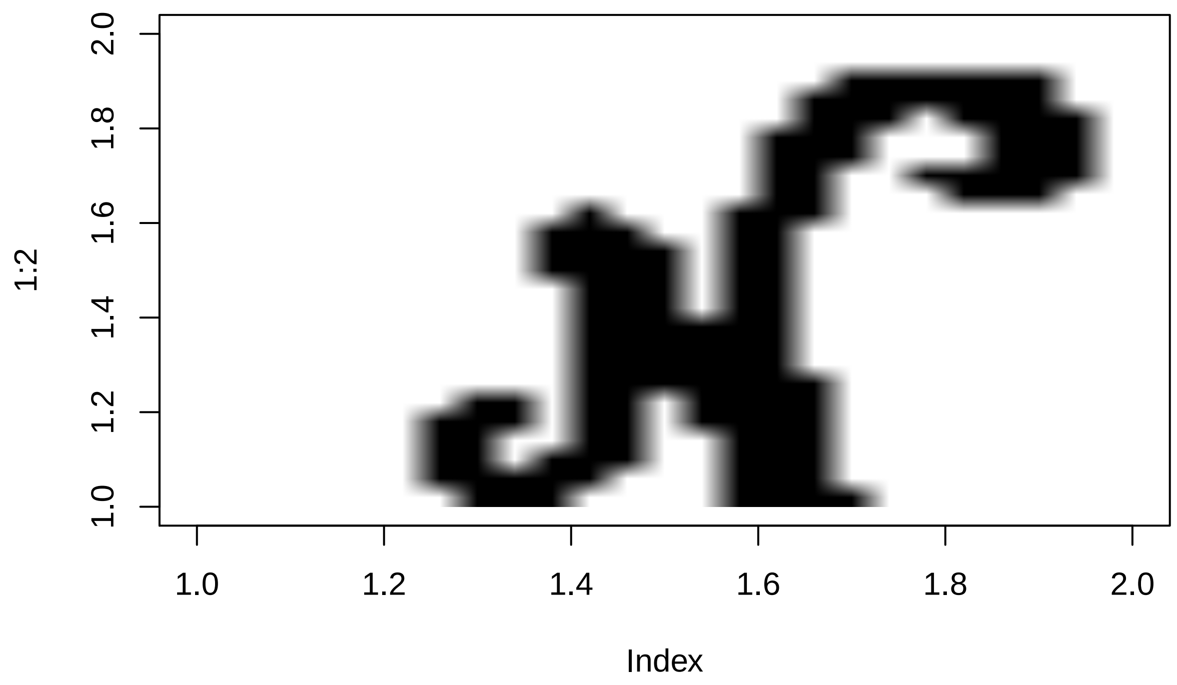
```r
##Below code block does the same for font N(second class).

Nfilenames = list.files(NPath)
# par(mfrow=c(4,4))
for(i in Nfilenames[1:length(Nfilenames)]){
  txt <- paste(NPath,i,sep="")
  tst <- readJPEG(txt)
  tst <- resizeImage(tst, w = 25, h = 25)
  tst <- ifelse(tst>0.5,1,0)
  tst <- tst[,,1]
  plotImage(tst)
  tst <- ramify::flatten(tst)
  tst<-c(0,tst)
  dataset<- rbind(dataset,tst)
}
```

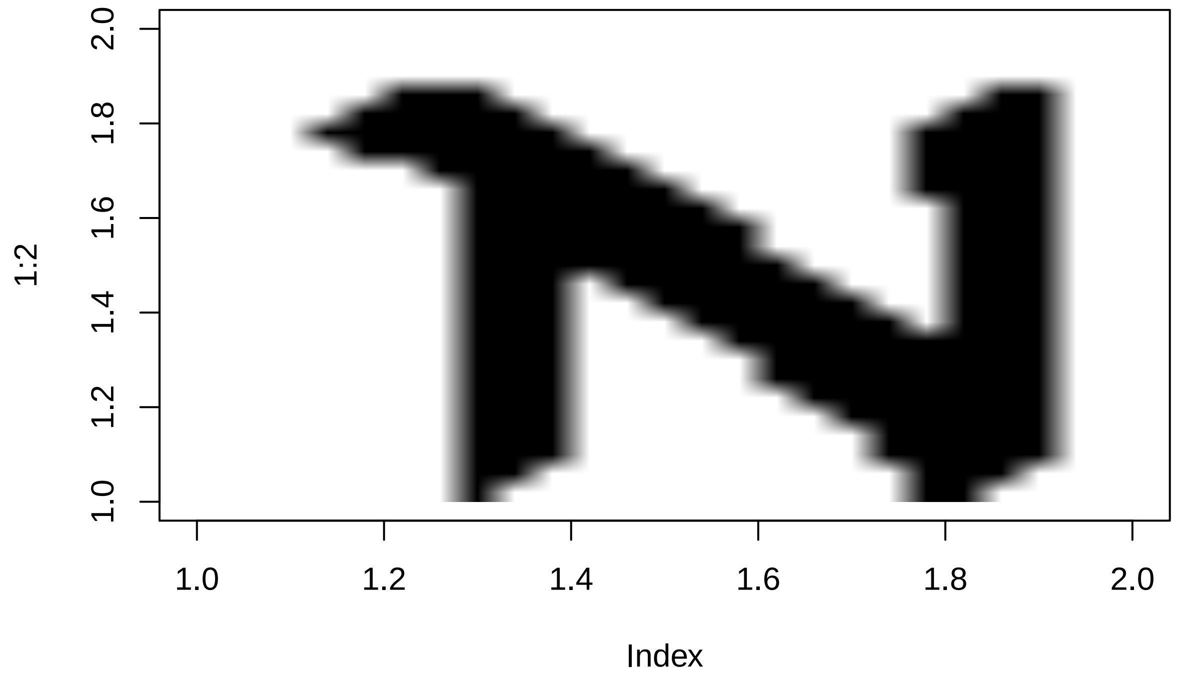

1:2
2.0
1.8
1.6
1.4
1.2
1.0
Index

1:2
2.0
1.8
1.6
1.4
1.2
1.0
1.0
1.2
1.4
1.6
1.8
2.0
Index

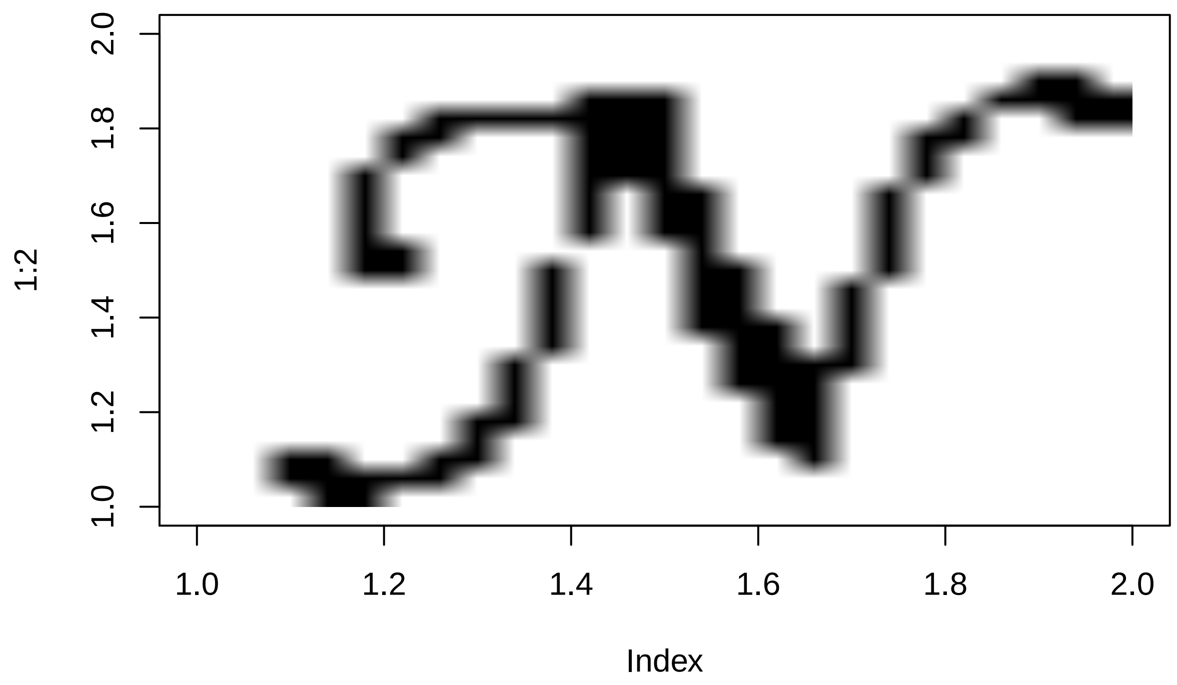

2.0
1.8
1.6
1.4
1.2
1.0
1:2
1.0
1.2
1.4
1.6
1.8
2.0
Index


1:2
2.0
1.8
1.6
1.4
1.2
1.0
1.0
1.2
1.4
1.6
1.8
2.0
Index

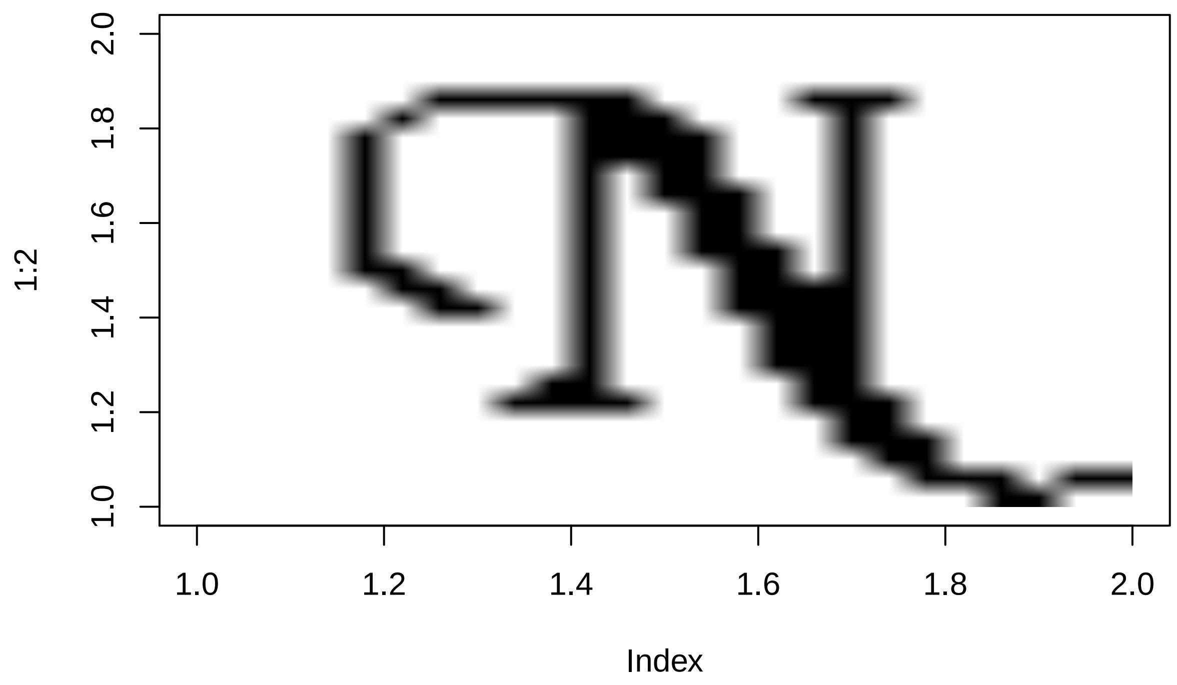

1.0
1.2
1.4
1.6
1.8
2.0
1:2
Index

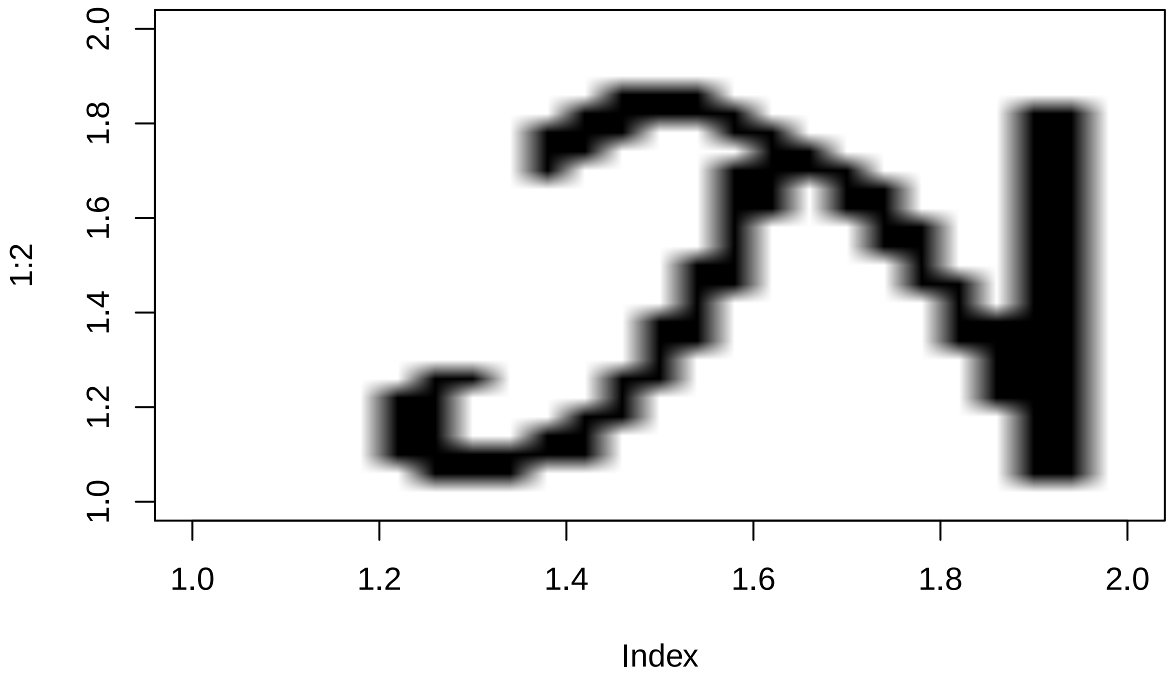


2.0
1.8
1.6
1.4
1.2
1.0
1:2
1.0
1.2
1.4
1.6
1.8
2.0
Index

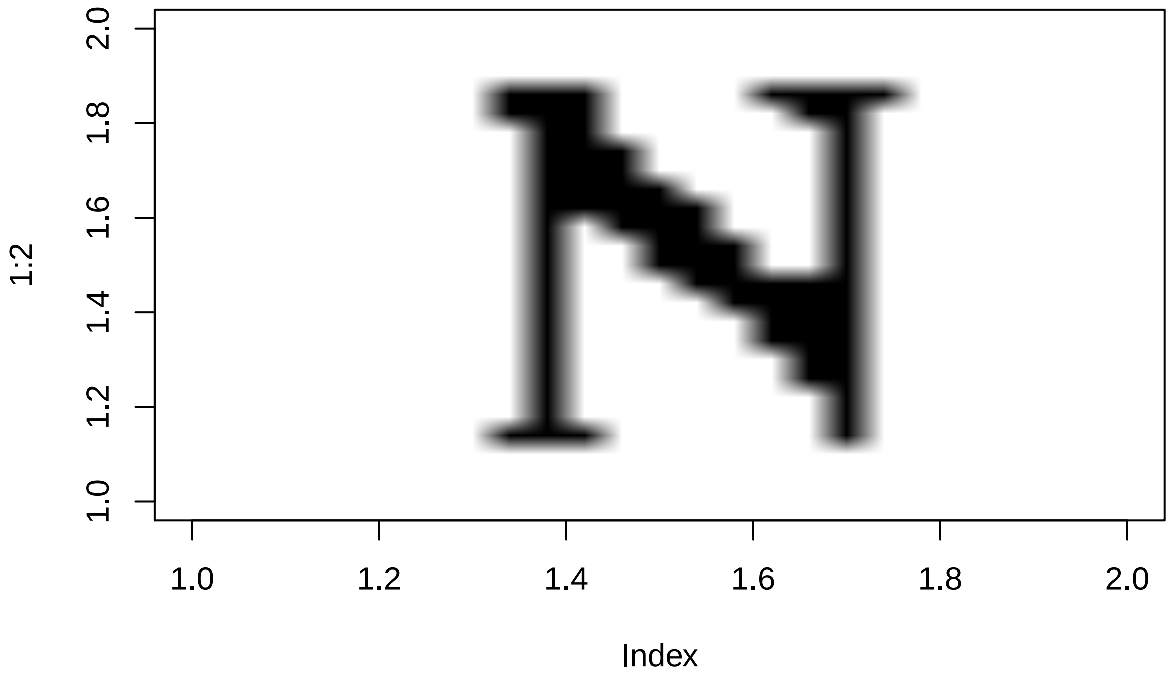


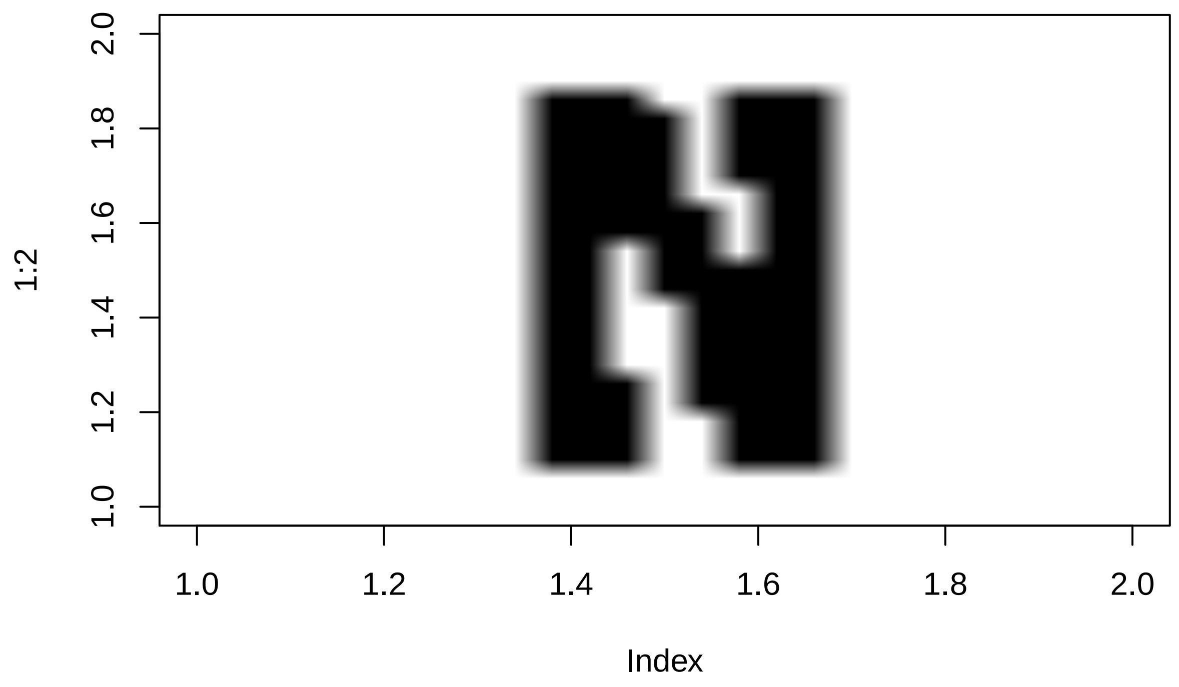
1:2
2.0
1.8
1.6
1.4
1.2
1.0
1.0
1.2
1.4
1.6
1.8
2.0
Index

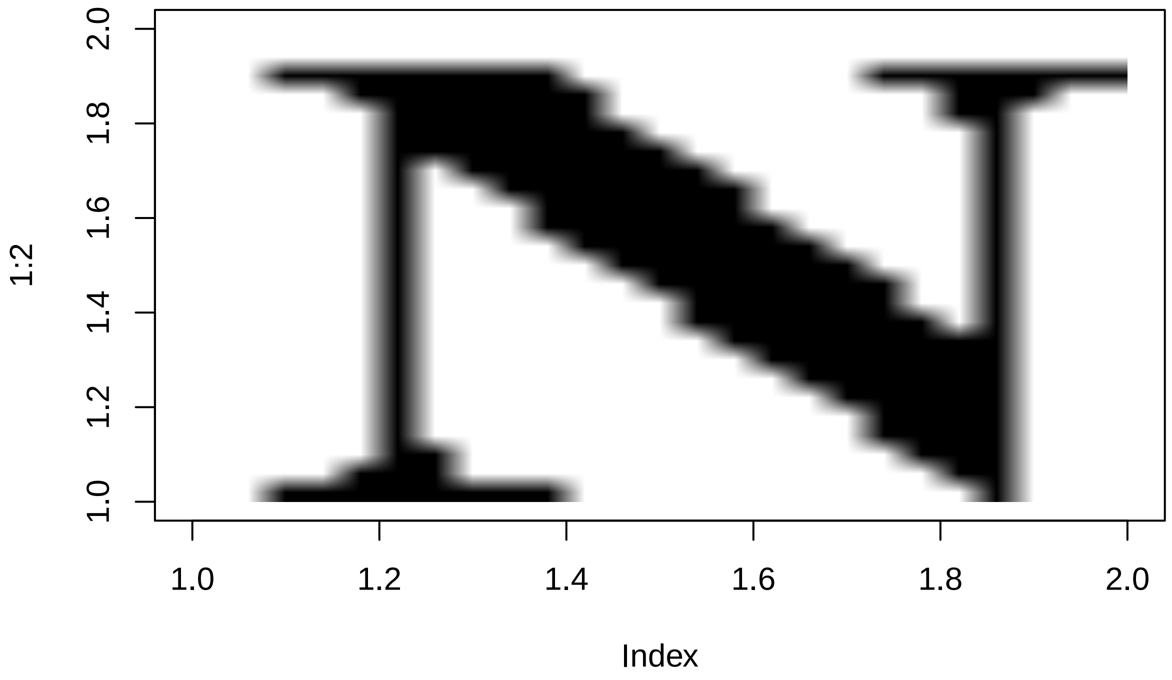

2.0
1.8
1.6
1.4
1.2
1.0
1:2
1.0
1.2
1.4
1.6
1.8
2.0
Index

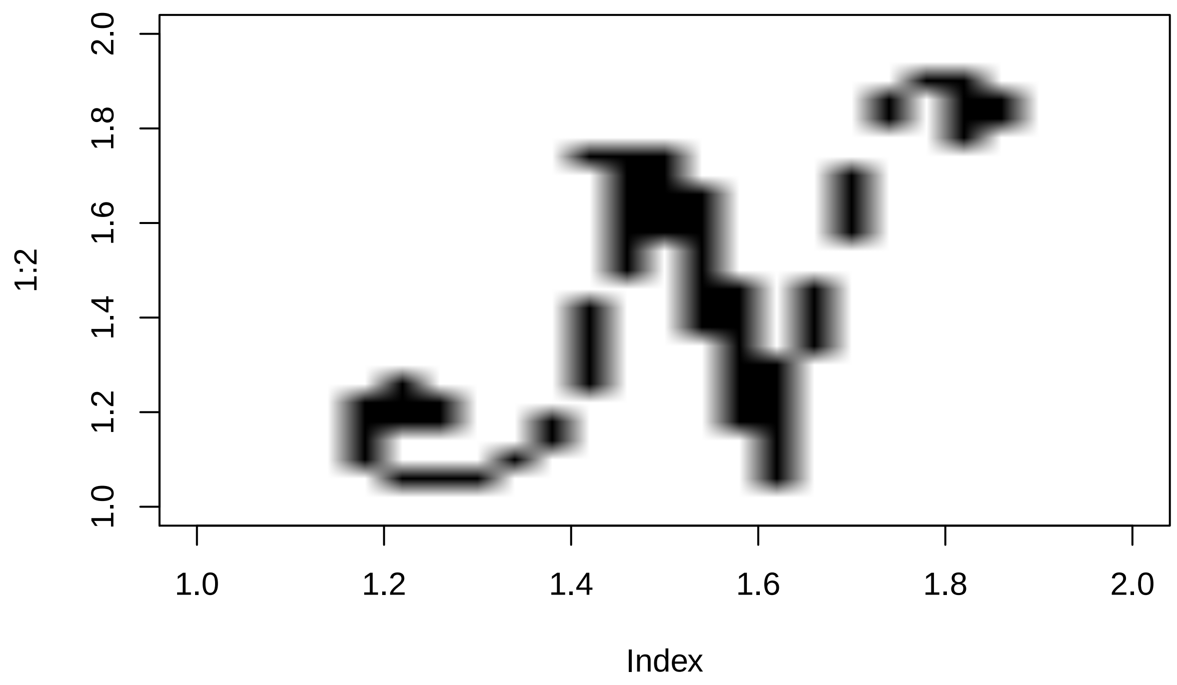


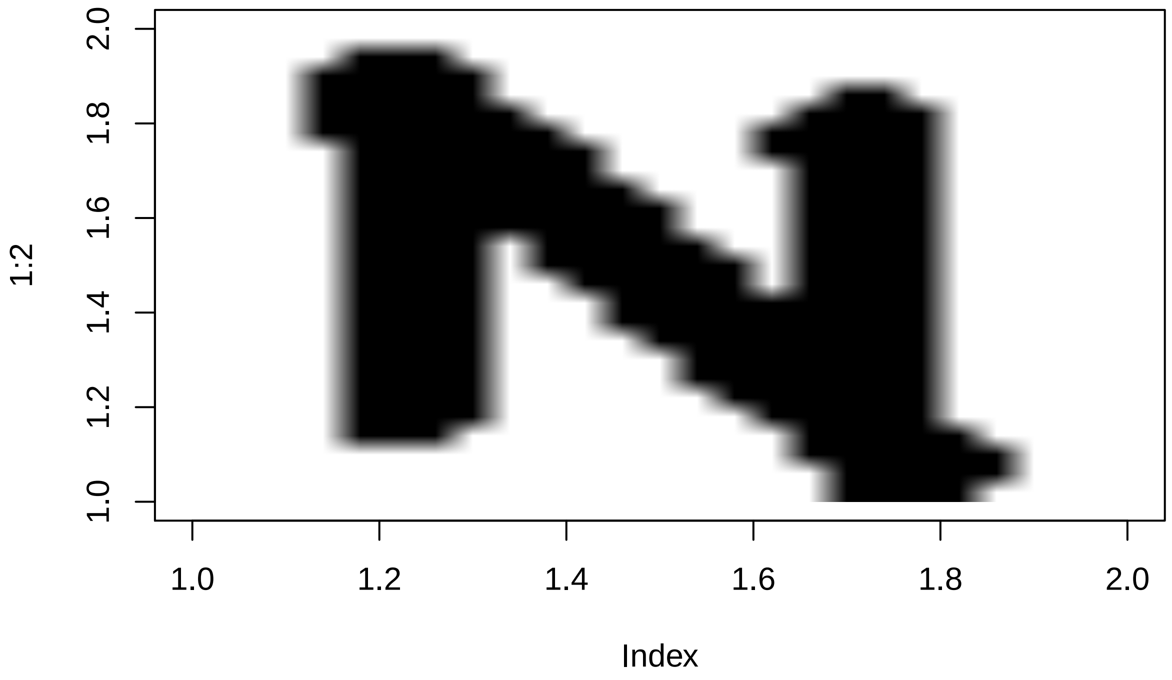
2.0
1.8
1.6
1.4
1.2
1.0
1:2
1.0
1.2
1.4
1.6
1.8
2.0
Index

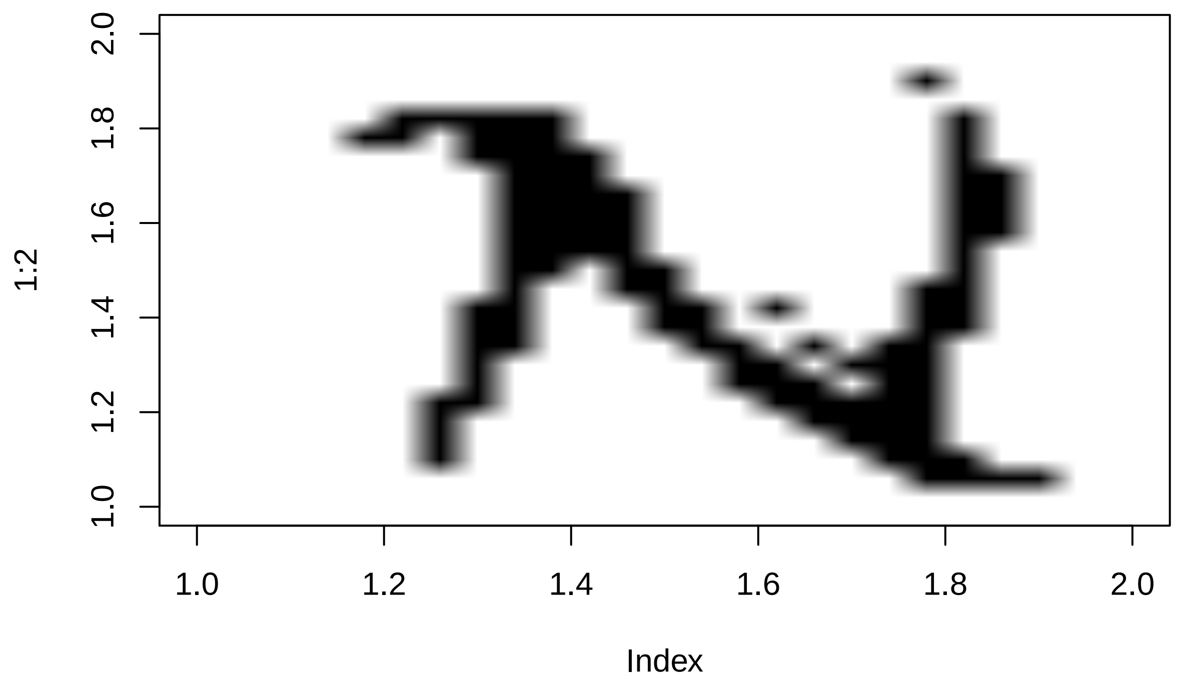

1:2
2.0
1.8
1.6
1.4
1.2
1.0
1.0
1.2
1.4
1.6
1.8
2.0
Index


2.0
1.8
1.6
1.4
1.2
1.0
1:2
1.0
1.2
1.4
1.6
1.8
2.0
Index

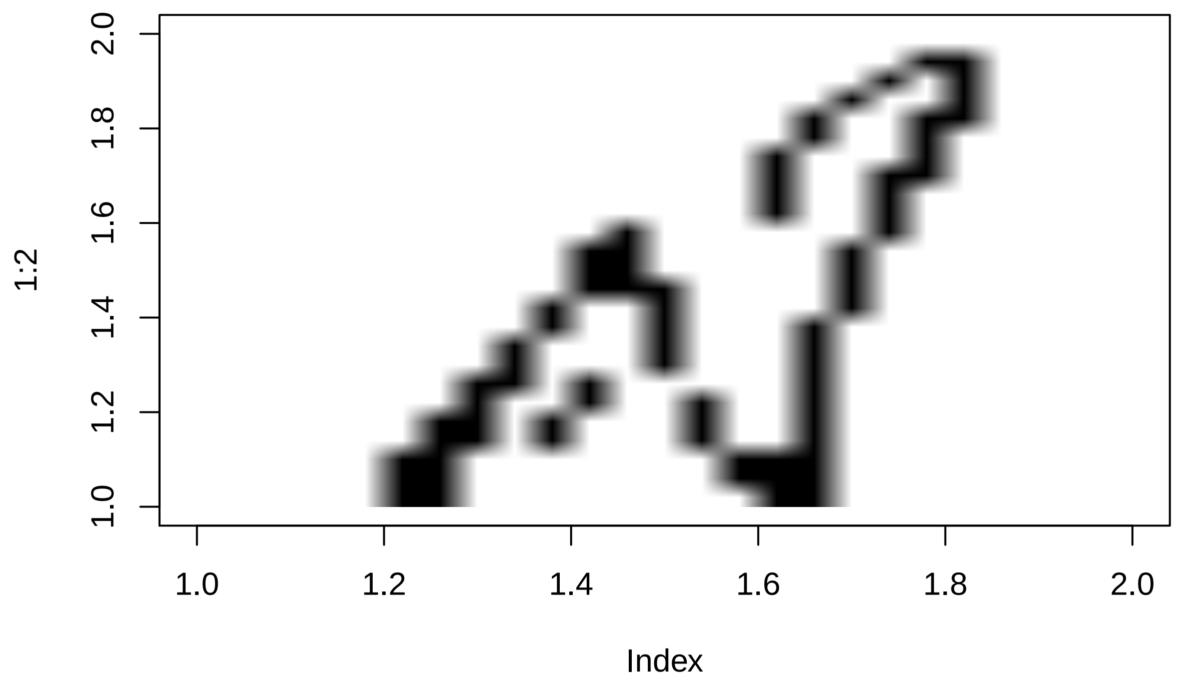

2.0
1.8
1.6
1.4
1.2
1.0
1:2
1.0
1.2
1.4
1.6
1.8
2.0
Index


1:2
2.0
1.8
1.6
1.4
1.2
1.0
1.0
1.2
1.4
1.6
1.8
2.0
Index

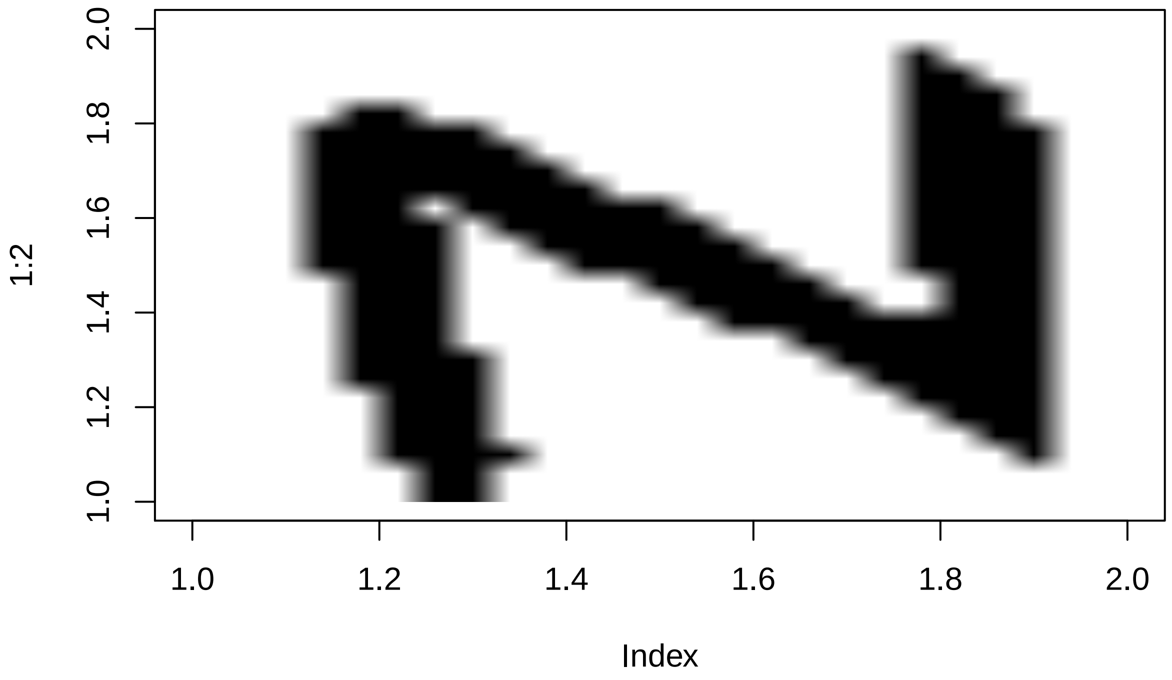

1:2
1.0
1.2
1.4
1.6
1.8
2.0
Index

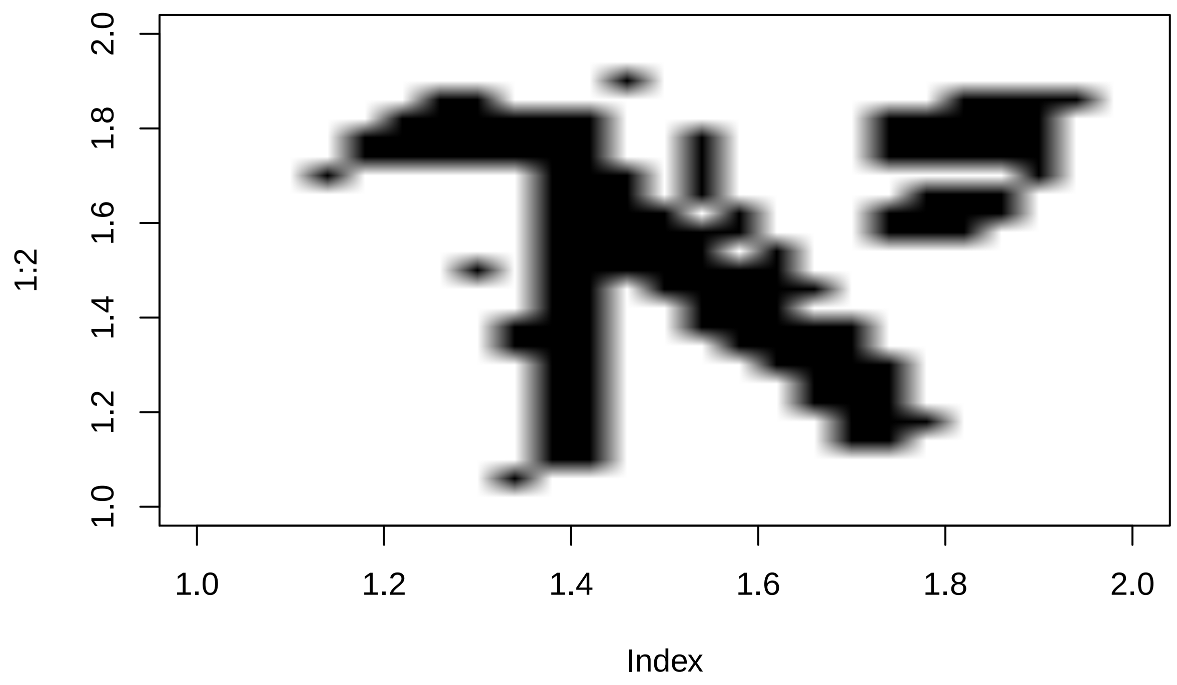


1:2
2.0
1.8
1.6
1.4
1.2
1.0
1.0
1.2
1.4
1.6
1.8
2.0
Index

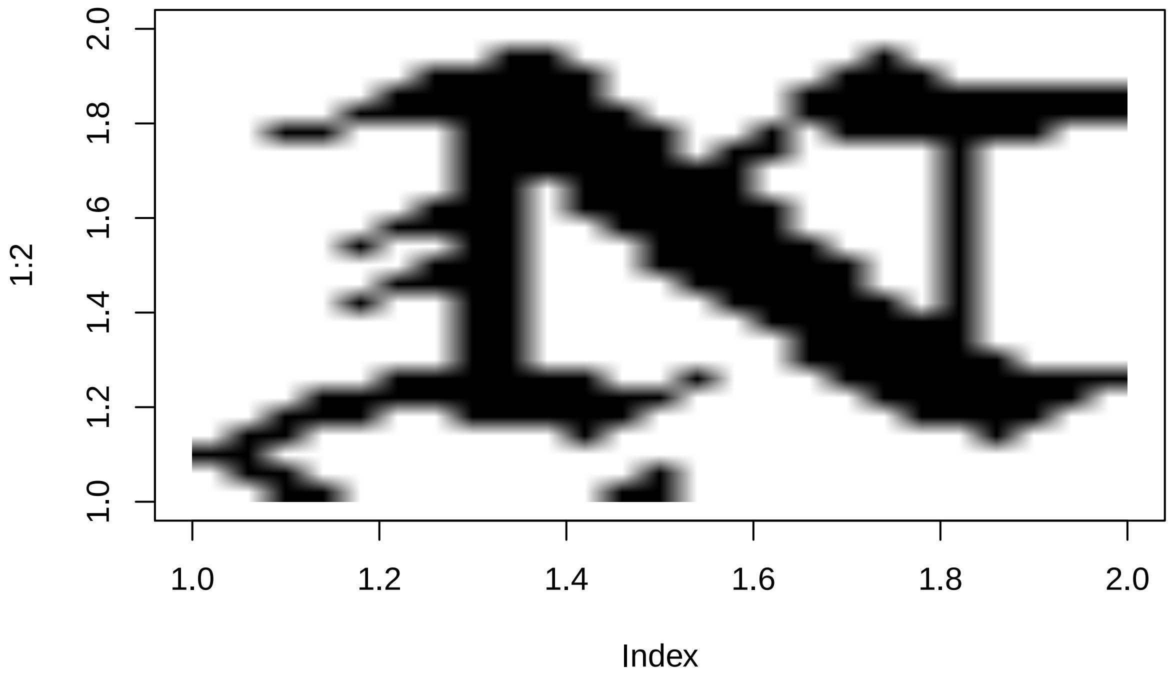


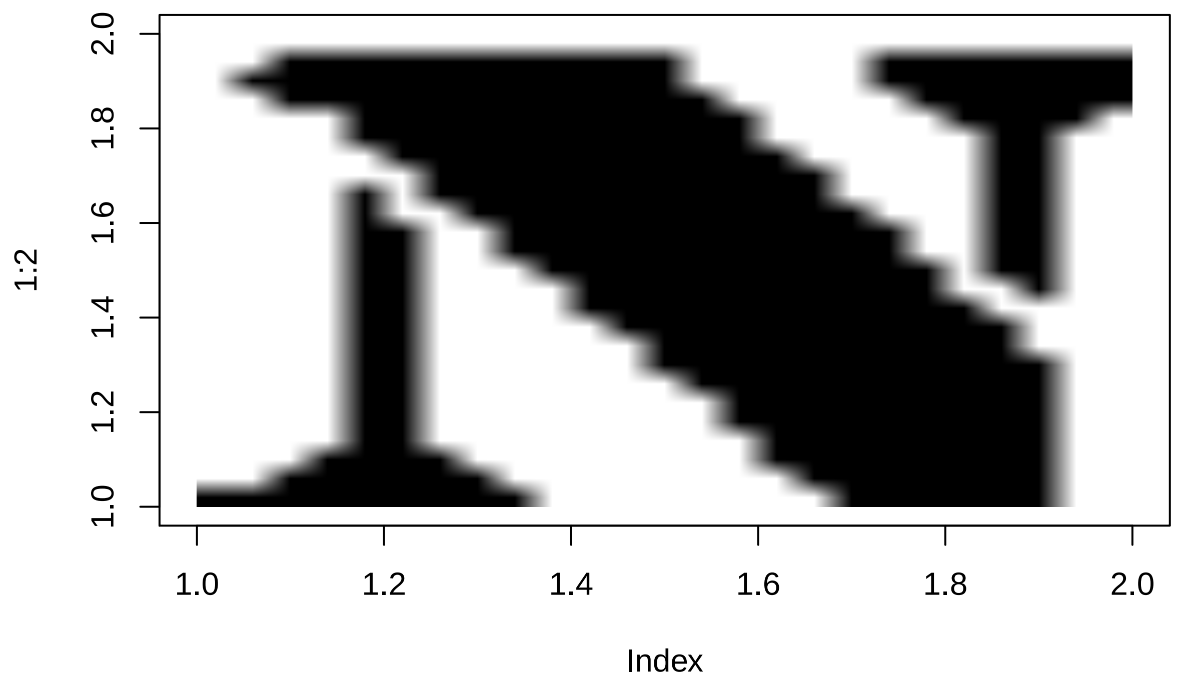
2.0
1.8
1.6
1.4
1.2
1.0
1:2
1.0
1.2
1.4
1.6
1.8
2.0
Index


1:2
2.0
1.8
1.6
1.4
1.2
1.0
1.0
1.2
1.4
1.6
1.8
2.0
Index



2.0
1.8
1.6
1.4
1.2
1.0
1:2
1.0
1.2
1.4
1.6
1.8
2.0
Index


1:2
Index
1.0
1.2
1.4
1.6
1.8
2.0



```
View(dataset)

########################
##Splitting data into test and train.
library(caTools)
```

```
## Warning: package 'caTools' was built under R version 4.0.3
```

```
data1 = sample.split(dataset,SplitRatio=0.6)
train = na.omit(subset(dataset,data1==TRUE),header=FALSE)
test = na.omit(subset(dataset,data1==FALSE),header=FALSE)


#########################
library(neuralnet)
```

```
## Warning: package 'neuralnet' was built under R version 4.0.4
```

```
set.seed(333)

#NeuralNetwork 1 using "rprop+" algorithm with 5 hidden layers
#2 repetitions, error metric "cross-entropy"
n <- neuralnet(X1~.,
               data = train,
               hidden = 5,
               err.fct = "ce",
               linear.output = FALSE,
               lifesign = 'full',
               rep = 2,
               algorithm = "rprop+",
               stepmax = 100000)
```

```
## hidden: 5 thresh: 0.01 rep: 1/2 steps:

##      54  error: 0.0225   time: 0.09 secs
## hidden: 5    thresh: 0.01    rep: 2/2    steps:      47  error: 0.01939  time: 0.03 secs
```

```
output <- neuralnet::compute(n,rep=2, test)
p1<- output$net.result
pred1 <- ifelse(p1 > 0.5, 1, 0)
tab1<-table(pred1,test$X1)
tab1
```

```
##
## pred1 0 1
##     0 4 0
##     1 7 9
```

```
rprop1 = paste("Accuracy: ",sum((pred1==test$X1))/length(test$X1) * 100,"%")
rprop1
```

```
## [1] "Accuracy:  65 %"
```

```
##############################
#NeuralNetwork 1 using "rprop+" algorithm with 10 hidden layers
#2 repetitions, error metric "cross-entropy"
n <- neuralnet(X1~.,
               data = train,
               hidden = 10,
               err.fct = "ce",
               linear.output = FALSE,
               lifesign = 'full',
               rep = 2,
               algorithm = "rprop-",
               stepmax = 100000)
```

```
## hidden: 10    thresh: 0.01    rep: 1/2    steps:      42 error: 0.00897  time: 0.03 secs
## hidden: 10    thresh: 0.01    rep: 2/2    steps:      33 error: 0.01663  time: 0.03 secs
```

```
output <- neuralnet::compute(n,rep=2, test)
p1<- output$net.result
pred1 <- ifelse(p1 > 0.5, 1, 0)
tab2<-table(pred1,test$X1)
tab2
```

```
##
## pred1 0 1
##     0 4 2
##     1 7 7
```

```
rprop2 = paste("Accuracy: ",sum((pred1==test$X1))/length(test$X1) * 100,"%")
rprop2
```

```
## [1] "Accuracy:  55 %"
```

```
#############################
#NeuralNetwork 1 using "rprop+" algorithm with 10 hidden layers
#2 repetitions, error metric "cross-entropy"
n <- neuralnet(X1~.,
               data = train,
               hidden = 10,
               err.fct = "ce",
               linear.output = FALSE,
               lifesign = 'full',
               rep = 2,
               algorithm = "sag",
               stepmax = 100000)
```

```
## hidden: 10    thresh: 0.01    rep: 1/2    steps:      41 error: 0.01284  time: 0.05 secs
## hidden: 10    thresh: 0.01    rep: 2/2    steps:      27 error: 0.01699  time: 0.03 secs
```

```
output <- neuralnet::compute(n,rep=2, test)
p1<- output$net.result
pred1 <- ifelse(p1 > 0.5, 1, 0)
tab3<-table(pred1,test$X1)
tab3
```

```
##
## pred1 0 1
##     0 4 2
##     1 7 7
```

```
sag = paste("Accuracy: ",sum((pred1==test$X1))/length(test$X1) * 100,"%")
sag
```

```
## [1] "Accuracy:  55 %"
```

```
###########################
#NeuralNetwork 1 using "backprop" algorithm with 5 hidden layers
#1 repetitions, error metric "cross-entropy"
#learning rate =1e-1
n <- neuralnet(X1~.,
               data = train,
               hidden = 5,
               err.fct = "ce",
               linear.output = FALSE,
```

```r
               lifesign = 'full',
               rep = 1,
               algorithm = "backprop",learningrate = 1e-1,
               )
```

```
## hidden: 5    thresh: 0.01    rep: 1/1    steps:    1000  min thresh: 0.0110599256766222
##                                                     1097  error: 8.71215  time: 0.51 secs
```

```r
output <- neuralnet::compute(n,rep=1, test)
p1<- output$net.result
pred1 <- ifelse(p1 > 0.5, 1, 0)
tab4<-table(pred1,test$X1)
tab4
```

```
##
## pred1 0 1
##     0 4 3
##     1 7 6
```

```r
backprop1 = paste("Accuracy: ",sum((pred1==test$X1))/length(test$X1) * 100,"%")
backprop1
```

```
## [1] "Accuracy:  50 %"
```

```r
###########################

#NeuralNetwork 1 using "backprop" algorithm with 15 hidden layers
#1 repetitions, error metric "cross-entropy"
#learningrate = 1e-1
n <- neuralnet(X1~.,
               data = train,
               hidden = 15,
               err.fct = "ce",
               linear.output = FALSE,
               lifesign = 'full',
               rep = 1,
               algorithm = "backprop",learningrate = 1e-1,
)
```

```
## hidden: 15    thresh: 0.01    rep: 1/1    steps:    1000 min thresh: 1.39086166097821
##                                                      2000 min thresh: 1.34578734165831
##                                                      3000 min thresh: 0.181020143296068
##                                                      3531 error: 11.03737 time: 3.49 secs
```

```r
output <- neuralnet::compute(n,rep=1, test)
p1<- output$net.result
pred1 <- ifelse(p1 > 0.5, 1, 0)
tab5<-table(pred1,test$X1)
tab5
```

```
##
## pred1 0 1
##     0 3 0
##     1 8 9
```

```r
backprop2 = paste("Accuracy: ",sum((pred1==test$X1))/length(test$X1) * 100,"%")
backprop2
```

```
## [1] "Accuracy:  60 %"
```

```r
###########################
#NeuralNetwork 1 using "backprop" algorithm with 20 hidden layers
#1 repetitions, error metric "cross-entropy"
#learningrate = 1e-2
n <- neuralnet(X1~.,
               data = train,
               hidden = 20,
               err.fct = "ce",
               linear.output = FALSE,
               lifesign = 'full',
               rep = 3,
               algorithm = "backprop",learningrate = 1e-2,
)
```

```
## hidden: 20    thresh: 0.01    rep: 1/3    steps:    1000 min thresh: 0.114329348956314
##                                                     2000 min thresh: 0.0530034693273722
##                                                     3000 min thresh: 0.0380991897962844
##                                                     4000 min thresh: 0.029857040194649
##                                                     5000 min thresh: 0.0234239022808163
##                                                     6000 min thresh: 0.0191362189810728
##                                                     7000 min thresh: 0.0160738623735249
##                                                     8000 min thresh: 0.0130833411558113
##                                                     9000 min thresh: 0.0111149913600638
##                                                    10000 min thresh: 0.0104001657552924
##                                                    10389 error: 0.02747  time: 14.65 secs
## hidden: 20    thresh: 0.01    rep: 2/3    steps:    1000 min thresh: 0.118853510177578
##                                                     2000 min thresh: 0.074290409146465
##                                                     3000 min thresh: 0.0541510179166474
##                                                     4000 min thresh: 0.0440969222312907
##                                                     5000 min thresh: 0.0351894632045812
##                                                     6000 min thresh: 0.0293740397148467
##                                                     7000 min thresh: 0.0252982985754679
##                                                     8000 min thresh: 0.0222621636883188
##                                                     9000 min thresh: 0.0199003545383063
##                                                    10000 min thresh: 0.0180047963039254
##                                                    11000 min thresh: 0.0164469860386027
##                                                    12000 min thresh: 0.0151425465924176
##                                                    13000 min thresh: 0.0140334607959823
##                                                    14000 min thresh: 0.0130783796189994
##                                                    15000 min thresh: 0.0122469698742178
##                                                    16000 min thresh: 0.0115164410367468
##                                                    17000 min thresh: 0.0108693199827121
##                                                    18000 min thresh: 0.0102919754974471
##                                                     18551 error: 0.04157  time: 24.03 secs
## hidden: 20    thresh: 0.01    rep: 3/3    steps:     1000 min thresh: 0.0816272102981028
##                                                      2000 min thresh: 0.0248573511584979
##                                                      3000 min thresh: 0.016932539888543
##                                                      4000 min thresh: 0.0132242901035435
##                                                      4984 error: 0.03523  time: 5.91 secs
```

```
output <- neuralnet::compute(n,rep=2, test)
p1<- output$net.result
pred1 <- ifelse(p1 > 0.5, 1, 0)
tab6<-table(pred1,test$X1)
tab6
```

```
##
## pred1  0  1
##     0  1  2
##     1 10  7
```

```
backprop3= paste("Accuracy: ",sum((pred1==test$X1))/length(test$X1) * 100,"%")
backprop3
```

```
## [1] "Accuracy:  40 %"
```

```
###########################
tab1
```

```
##
## pred1 0 1
##     0 4 0
##     1 7 9
```

```
rprop1
```

```
## [1] "Accuracy:  65 %"
```

```
tab2
```

```
##
## pred1 0 1
##     0 4 2
##     1 7 7
```

```
rprop2
```

```
## [1] "Accuracy:  55 %"
```

```
tab3
```

```
##
## pred1 0 1
##     0 4 2
##     1 7 7
```

```
sag
```

```
## [1] "Accuracy:  55 %"
```

```
tab4
```

```
##
## pred1 0 1
##     0 4 3
##     1 7 6
```

```
backprop1
```

```
## [1] "Accuracy:  50 %"
```

```
tab5
```

```
##
## pred1 0 1
##     0 3 0
##     1 8 9
```

```
backprop2
```

```
## [1] "Accuracy:  60 %"
```

```
tab6
```

```
##
## pred1  0  1
##     0  1  2
##     1 10  7
```

```
backprop3
```

```
## [1] "Accuracy:  40 %"
```